Characterizing Dynamic Optimization Benchmarks for the Comparison of Multi-Modal Tracking Algorithms

Ron Bond

Department of Computer Science

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science (Computer Science)

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

Abstract

Population-based metaheuristics, such as particle swarm optimization (PSO), have been employed to solve many real-world optimization problems. Although it is often sufficient to find a single solution to these problems, there does exist those cases where identifying multiple, diverse solutions can be beneficial or even required. Some of these problems are further complicated by a change in their objective function over time. This type of optimization is referred to as dynamic, multi-modal optimization. Algorithms which exploit multiple optima in a search space are identified as niching algorithms. Although numerous dynamic, niching algorithms have been developed, their performance is often measured solely on their ability to find a single, global optimum. Furthermore, the comparisons often use synthetic benchmarks whose landscape characteristics are generally limited and unknown.

This thesis provides a landscape analysis of the dynamic benchmark functions commonly developed for multi-modal optimization. The benchmark analysis results reveal that the mechanisms responsible for dynamism in the current dynamic benchmarks do not significantly affect landscape features, thus suggesting a lack of representation for problems whose landscape features vary over time. This analysis is used in a comparison of current niching algorithms to identify the effects that specific landscape features have on niching performance. Two performance metrics are proposed to measure both the scalability and accuracy of the niching algorithms. The algorithm comparison results demonstrate the algorithms best suited for a variety of dynamic environments. This comparison also examines each of the algorithms in terms of their niching behaviours and analysing the range and trade-off between scalability and accuracy when tuning the algorithms respective parameters. These results contribute to the understanding of current niching techniques as well as the problem features that ultimately dictate their success.

Acknowledgements

I would like thank the individuals who helped me accomplish this thesis:

- My supervisor, Professor Beatrice Ombuki-Berman for her instruction and guidance;
- Professor Andries Engelbrecht for his insight in the field of optimization and for giving the work a much needed focus;
- Professors Michael Winter and Brian Ross for their feedback as members of the committee;
- Filipe Nepomuceno and Cale Fairchild for aiding in the deployment of experimentation;
- Bennie Leonard for the countless discussions and feedback during the planning stages of the work;
- Katrina Kroeze for her support, encouragement, and patience throughout;
- My parents for their continued support of my seemingly-endless academic career.

R.H.B

Contents

Chapter 1

Introduction

Computational Intelligence (CI) is a term used to describe the set of nature-inspired approaches to complex problems whose properties do not lend themselves to traditional methods such as statistical modelling [28]. These properties include, but are not limited to, noise, uncertainty, incomplete information, computing intractability, and dynamics. These properties pose issues for traditional modelling because they invalidate assumptions on which the modelling is based. Swarm Intelligence (SI) is a branch of CI that focuses on the social interaction of simple agents to form a more complex behaviour, generally unknown to the individual agent [28]. One such SI optimization method is named the Particle Swarm Optimization (PSO) [9] which was designed for continuous-space optimization.

Algorithms such as PSO were originally developed to converge eventually to a single, potentially optimal solution. Although it is often sufficient to find a single solution to an optimization problem, there exists cases where finding multiple differing candidates can be beneficial or even a requirement. An example of such a case is the training of neural network ensembles. A neural network is a machine learning technique used for pattern recognition and data classification tasks where the patterns are complex and non-linear [32]. Neural network ensembles are often used in replacement of a single network in order to provide classification from multiple different perspectives. In order to achieve this difference in perspective, it is important that the trained weight-configurations of the networks be significantly different between members of the ensemble [32]. A PSO algorithm used to train neural network ensembles should therefore yield candidate solutions from multiple areas of the search space; even if some of those areas are sub-optimal in comparison to the best area found. Search spaces where multiple optima or near-optima exist are often referred

as being multi-modal and can present additional challenges, even for algorithms that do not attempt to find more than one solution. This is true because algorithms are attracted to sub-optimal regions of the search space which may pull them away from global optima.

Another challenge to consider in optimization is the possibility of change. Static environments are those whose solution-space, also referred to as a landscape, remain unchanged over the course of optimization. Conversely, dynamic environments are those landscapes that do change while an algorithm is searching.

1.1 Overview

A dynamic environment can be described as a discrete series of static environments or states. These states are separated by change events, which in turn, are described as a transformation function which maps a previous state to its successive state. Describing dynamic landscapes in such a way highlights the inherited complexities of having to deal with multiple possible fitness landscapes using a single algorithm. Fitness landscapes are generally classified by means of identifying and measuring characteristics attributed to problem hardness [42]. These characteristics are reviewed in Chapter 2 along with their affects on optimization algorithms. Although landscape features are studied significantly in static landscape research, they are often ignored in dynamic analysis.

In order to emphasize the lack of consideration for landscape features in current dynamic research, this work will first evaluate current dynamic benchmark functions, namely the Moving Peaks Benchmark (MPB) [6, 7] and the Generalized Dynamic Benchmark Generator (GDBG) [24, 25]. A suite of landscape metrics introduced in [42] is used to accomplish the analysis; a review of these metrics is also included in chapter 2. This analysis serves to classify the dynamic benchmark functions according to each landscape feature, as well as to determine the variance and range of these measurements caused by dynamism.

A review of the current multi-modal PSO variants follows the benchmark analysis. Although similar comparative studies have been conducted in the past [18, 22], their analysis is limited in terms of the benchmarks and performance metrics used. Additionally, the experimentation does not provide empirical support for the parameter

configurations used for each of the algorithms, meaning the comparison is potentially unfair. The niching techniques used in the algorithms reviewed include speciation [15, 16], clustering [19], and predefined muti-swarm [1, 2, 3]. The analysis will also cover some adaptive variants of these techniques [4, 17, 18]. The term niching refers to an algorithm's ability to identify multiple optima [32]. The various algorithms are analyzed and compared based on their behaviours of maintaining multiple, distinct sub-populations. The algorithms are implemented and their niching performance compared in terms of accuracy and efficiency on the reviewed benchmark functions. A range of configurations are tested for each algorithm in order to provide a fair performance comparison of the PSO-variants under the different landscape characteristics observed from the benchmark analysis.

1.2 Objectives

The primary objective of this study is to review current benchmarks for continuous, dynamic, multi-modal optimization. As mentioned above, the experimentation uses various landscape analysis metrics to categorize the benchmarks based upon the metrics obtained from the environments generated. An examination of the variability on landscape characteristics caused by the change operators is also performed. The intention is to test the assumption that current benchmark change operators do not significantly vary landscape characteristics. This assumption is based on the absence of consideration to landscape structure in the current definition of dynamic environments.

A secondary objective is to perform an empirical analysis of the different niching algorithms which identify and exploit multiple optima in dynamic environments. This analysis is conducted against the categorized benchmark functions from the primary objective in order to study the performance of the multi-modal techniques under different landscape characteristics. These research goals can be further summarized by the following:

- Survey continuous-landscape analysis techniques;
- Survey and implement current benchmark functions used for dynamic, continuous, multi-modal environments;
- Characterize these benchmark functions according to the landscape analysis characteristics;

- Survey and implement current algorithms designed for dynamic, multi-modal optimization; and
- Provide an empirical analysis and comparison of the algorithms' abilities to locate/track multiple optima under varying landscape conditions/characteristics.

1.3 Contributions

The contributions made are:

- A classification of the current dynamic benchmark generators based on landscape analysis characteristics;
- Proposed enhancements to the definition of change in the context of dynamic environments;
- A statistical comparison of the niching algorithms under each of the benchmark functions; and
- The implementation of the various algorithms/benchmarks reviewed for both static and dynamic multi-modal environments in the open-source library, CILib [31].

1.4 Thesis Outline

Chapter 2 provides background information for dynamic environments, landscape analysis and metrics, as well as formal definitions of the Moving Peaks Benchmark (MPB) [6, 7] and the Generalized Dynamic Benchmark Generator (GDBG) [24, 25] generator functions.

Chapter 3 presents the experimentation and results of the benchmark categorization based on landscape analysis techniques from the previous chapter.

Chapter 4 presents the original Particle Swarm Optimization algorithm [9]. A detailed description, along with the advantages and disadvantages of the algorithm in the context of dynamic environments are discussed.

Chapter 5 provides a summary of each of the niching PSO operators. Finally, an analysis is performed to highlight the potential strengths and weaknesses of these operators in comparison to one another.

Chapter 6 provides empirical results of the niching PSO algorithms on the benchmark generator functions from chapter 2. Various algorithm parameter configurations are tested in order identify ideal configurations per benchmark function. The algorithms are then statistically tested against one another using the best respective configurations found under each function.

Chapter 7 summarizes the results and conclusions of the research. The closing chapter also makes recommendations for potential directions of future work with regards to dynamic benchmark functions, adaptive algorithm behaviours, as well as niching performance metrics.

Chapter 2

Landscape Analysis, Measurements, and Benchmarks

This chapter describes the representation of optimization problems, commonly referred to as a fitness landscape. An overview of landscape features that are often associated with optimization performance is also included. Finally, a list of measurements aimed at quantifying the landscape features is described. The list of features below does not cover all possible landscape characteristics, but only those that are generally applicable and measurable in continuous space are considered.

2.1 Fitness Landscapes

In order to employ a meta-heuristic to a given problem, a representation of a feasible solution must first be defined for the problem. This representation can be described as an aggregation of one or several values representing the input variables of the problem. The next requirement is the definition of a fitness function: a function that takes as input a solution representation and outputs a corresponding fitness value. The returned value(s) is thus a solution's rating based on the objective(s) of the problem.

Using the notions of solution representations and fitness values, a fitness landscape can be conceptualized as the topography generated by using the solutions as spatial positions and their corresponding fitness values as the value of the surface. As stated in [41], it is a common misconception in literature that the fitness landscape is a sole product of the defined fitness function. Although the fitness function does provide the values for a given landscape, the landscape itself is defined by the distribution of fitness

values. This distribution is defined by the notions of distance and neighbourhoods which differ depending on the operator(s) used by an algorithm [41]. A landscape is the product of a particular algorithm's experience. The distinction between fitness function and landscape is an important contribution made in the attempts at fitness landscape formalization in the works of Wright [52], Jones [53], and Sadler [54]. In other words, a landscape is formed by the solutions/positions used in generating it. Since the solutions considered are ultimately decided by the operators of the optimization algorithm, the resulting landscape will be different for each algorithm, even with a common fitness function.

2.2 Landscape Features

The following section highlights some general features of landscapes that are believed to affect search performance. The features listed below do not include those that are specific to any algorithm. Instead, only those features that apply to meta-heuristics in general are considered. It should be noted that the generality of these landscape features does not suggest that they affect all algorithms equally, but that their effects apply to most optimization algorithm to some degree.

2.2.1 Modality

The term modality refers to the number of local optima in a landscape. Local optima are defined in [55] as being positions or regions of a distribution with a better fitness value than all of its neighbouring positions. Local optima are often considered to be a problem since they tend to attract the search towards them and away from global optima. Common features of a multi-modal landscape include the number and distribution of local optima, as well as their basin size. The term basin in this context refers to the basin of attraction of optima, which describes the extent or coverage of the landscape where optima attract search [53]. Having many local optima can significantly reduce the chance of the search finding a global optimum; similarly local optima with large basins of attraction, relative to those of global optima, can also reduce the chance of finding the best area.

2.2.2 General Searchability

Landscape searchability [42] describes an algorithm's ability to infer a new position from the current position, where the new position has a better fitness than the current

position. Searchability is a generalization of the term evolvability [48], which deals specifically with this attribute pertaining to evolutionary algorithms. Although, by definition, searchability is highly dependant on the searching algorithm's operators, a generalized version of it can be described as the co-variance relationship between the spatial and fitness difference across positions in the landscape. This relationship is discussed further in the landscape metrics section below.

2.2.3 Ruggedness

The term ruggedness, in the context of optimization landscapes, refers to the distribution of local optima. High ruggedness in an area of the landscape can be described as having high variability between fitness values of neighbouring points. Rugged landscapes can have a negative effect on a search algorithm because they increases the likelihood of the algorithm settling on a local optimum due to inconsistent gradient information guiding the search [42].

2.2.4 Neutrality

Neutrality in a landscape refers to the distribution and size of regions where all positions have equal fitness values [49]. Equal in this context is relative to the scale of fitness values and therefore could be subject to a threshold ϵ [49]. Neutrality can cause similar issues as modality since these regions do not provide sufficient information or attraction to a useful direction. Furthermore, algorithms that use runtime statistics for adaptive behaviours may be misguided because the lack of change in the fitness evaluations can be mistaken for stagnation or convergence.

2.3 Landscape Dynamics

Static environments can be thought of as special cases of dynamic environments where the rate of change is zero. This suggests that dynamic environments share all of the same challenges of static environments, with the following additional complexities.

2.3.1 Change Characteristics

Currently defined change characteristics of dynamic environments deal with how and when the optima change positions throughout the landscape during optimization. A considerable amount of work is found in the literature on the characterization and

classification of change in dynamic environments. Eberhart *et al.* [34, 35] classifies changes in terms positions and fitnesses of global optima. Angeline's [36, 37] classification system uses the general trajectories of the moving optima. In [38], De Jong adds the concept of spatial and temporal severity to this trajectory classification. Temporal severity defines the time in which an algorithm has to optimize the current landscape before a change is invoked, while spatial severity outlines the movement of optima. In [39], Weicker's classification system uses direction, trajectory, spatial and temporal severities, as well as the spatial relation between movements (homogeneity).

In a review of the various dynamic classifications, a generalized framework is proposed by Duhain in [33]. The framework identifies an overlap between the previously mentioned classification systems, amalgamating them into a singular classification system. De Jong's spatial and temporal severities are re-defined as follows:

(Quasi)-Static is the set of dynamic problems whose spatial and temporal severities are either zero, a static environment, or some value too small to be considered relevant or truly dynamic.

Progressive changes are frequent but small relative the spatial size of the landscape.

Abrupt changes are infrequent but large relative the spatial size of the landscape.

Chaotic environments change frequently and their spatial shifts are large.

In addition to these four roughly-defined categories, Eberhart *et al.* [34, 35] and Angeline's classes [36, 37] are also used to create a 27-class system where all combinations of the three systems are considered. In terms of search algorithms, there has been little development in the exploitation of dynamic characteristics such as patterns in change in [39]. Instead, these characteristics are often disregarded in the development of optimization algorithms.

2.3.2 Landscape Regularity

Landscape regularity is a term introduced in this thesis which refers to the variability between the static landscapes of a dynamic environment with respect to landscape characteristics. This dynamic is often ignored in literature as characterization of dynamic optimization functions is usually based on the temporal and spatial severity, as seen in the previous section. Although altering the landscape in order to vary its

characteristics would be classified as a spatial alteration, it is unlikely that any of the spatial-change operators used in current benchmarks would have any significant effect on landscape features based on the operators' sole focus of altering position of optima. Landscape regularity is an important dynamic to consider since it is likely that there exists a class of real-world, dynamic problems in which the characteristics do vary significantly over time. This lack of consideration in the benchmark functions would also carry over to the design of search algorithms, since they are often used for competitive comparison in algorithms design literature. The issue that arise from varied landscape features is that a single algorithm must optimize multiple landscape types.

2.3.3 Change Predictability/Detectability

Changing landscapes present a variety of issues for different algorithms. Change prediction and change detection [1, 33] are mechanisms used to deal with these issues. Environments with a periodic change are trivial to predict [13], given that the change schedule is known in advance. Change detection requires re-evaluation of a previously evaluated solution in order to compare the two fitness values [1, 2]; if they are different then it might be evidence of a change. Such detection mechanisms are not guaranteed to work in environments where a change only affects a portion of the landscape, or if noise exists in the fitness function that cannot be identified. Algorithms that depend on change detection can suffer significantly if their detection mechanisms either fail to detect change or constantly detect false changes [18], e.g., mistaking noise for an environment change. The next section summarizes the metrics used to quantify the above landscape features.

2.4 Landscape Analysis Metrics

The specific implementation of landscape metrics used in this work are based on those proposed either in part or in whole by Malan in [42]. These metrics are aimed at measuring ruggedness, dispersion, gradients, and general searchability of the individual landscapes of the dynamic environment. This means that the metrics do not consider time and will yield their measurement for each static landscape in the series.

2.4.1 First Entropic Measure

The First Entropic Measure (FEM) [42] serves to measure ruggedness of a given landscape. The word entropy from the FEM refers to the level of uncertainty in a given sample. The sample in this measure is a set of fitness values taken by a random walk. Simple random walks are usually done by starting at a random position in the landscape and repeating the process of stepping in a random direction for the next sample. Vassilev *et al.* proposed this type of measure in [45, 46, 47] in an attempt to quantify neutrality and ruggedness. This method applied strictly to discrete landscapes; the FEM is Malan's [42] adaptation for the continuous space.

The random walk used by the FEM is called a progressive random walk. Simple random walks have the potential of sampling an unrepresentative portion since the direction of a step is completely random [42]. The progressive variety attempts to cover a more representative space by starting from an edge in the landscape and randomly stepping in a biased fashion toward the opposing edge. The step samples obtained by the walk are interpreted as three-point-objects where an object is made up of step $(x-1, x, x+1)$. These objects can be of a neutral, smooth, or rugged type. The types are decided based on the fitness values of the points relative to one another. For example an object is considered rugged if fitness-difference from the middle point to the other two differs in direction (+/-). The value returned by this metric is a floating-point value between 0 and 1 representing the average proportion of rugged points among the objects. A more detailed description of these types as well as the progressive random walk can be found in [42]. It should be noted that since this metric operates in continuous space, the equality of two points are based on an error threshold. Two values are considered equal if the difference between them is smaller than the threshold value.

The FEM is also sensitive to the size of step taken by the random walk, once again due to the continuous nature of the landscape. This is important to notice since certain functions may yield highly rugged measurements when smaller steps are taken and at the same time produce low ruggedness for larger steps. For this reason the step size used in this study, as well as those of Malan [42], ruggedness is measured at a step size of 1% and 10% of the landscape's domain. These values are also referred to as micro and macro scales respectively [42].

2.4.2 Dispersion Metric

The dispersion metric (DM) is a measure created by Lunacek and Whitley [50] and is aimed at distinguishing multi-funnel landscapes. A funnel, in this context, refers to is a basin shape on the macro-scale that consists of clustered local optima [40]. Multi-funnel landscapes can pose problems for algorithms that converge where time needed to decide which funnel holds the true global optima may be longer than the convergence speed of the algorithm allows. In other words, it is not guaranteed that a population finds the local optima of each funnel before its shrinking search radius converges on a single funnel.

The DM works by taking a uniform sample of the fitness landscape and comparing the dispersion amongst its points to the dispersion of a subset of those points. This subset is formed using a proportion of points with the best fitness values. Dispersion is simply the average distance between each pair of points in the set. The subset of points represents the peak of the funnel (whether maximizing or minimizing) and therefore should be closer to each other in a single funnel scenario. If however the dispersion of the subset is greater than the overall sample, then it can be reasonably assumed that the subset contains samples from different funnels within the landscape.

In an attempt to generalize the DM for the use on functions with varying domains, Malan [42] proposed that the samples be normalized in terms of their positions in space to a scale of 0 to 1 (for all dimensions). The threshold which decides inclusion for the subset of best solution is described as a top $p\%$ proportion of the sample.

2.4.3 Gradient Measures

The gradient measures (GM_{avg},GM_{max},GM_{std}) [42] like the FEM, uses neighbourhood information of a random walk in order to look at differences in fitness values. The gradient measure serves to quantify the magnitude of the difference without regard to direction. A reason to consider steepness of gradients is that it should highly, positively-correlate with the likelihood of deceptiveness. Deceptive landscapes are those with larger areas of misinformation, meaning that the information gathered by a particular algorithm in those areas would guide it away from the global optima instead of towards it. Deception, much like the landscape itself, is a combined product of the fitness function and the algorithm being applied. This means that the effect of gradients may not be the same for all optimization algorithms. Despite this, the

deception caused by large gradients does apply to population based algorithms with global attractors.

The GM uses a modified version of the progressive random walk (from the FEM), called a Manhattan progressive random walk [42], in order to guarantee a constant step size throughout the walk. This modification translates to selecting a random dimension of the landscape and only moving in that direction. This is done to avoid the computational complexities of assuring equal step sizes across potentially many dimensions.

The points sampled from the walk are transformed into individual gradient measures by:

$$g(t) = \frac{f(x(t+1)) - f(x(t))/(f^{max} - f^{min})}{s/(\sum_{i=1}^{n}(x_i^{max} - x_i^{min}))} \tag{2.1}$$

where $f(x)$ is the fitness function, $x(t)$ is the point sampled at step t, and s is the constant step size of the walk. The result of this transformation is a series of values representing the gradient difference between step x and $x+1$:

$$g(t), g(t+1), ..., g(t+(T-1)) \tag{2.2}$$

Note that in Equation 2.1, f^{min}, f^{max} as well as x_i^{min}, x_i^{max} represent the range of fitness values encountered in the walk and the range of the i^{th} dimension of the landscape domain. These values are used simply to normalize the gradient values so that fitness landscapes with similar structure share common gradient measurements regardless of fitness and domain scales.

The average, standard deviation, and maximum value are taken from the gradient series. The average is used in tandem with the other two as described below but can also be used to compare different landscapes with another due to the normalization described above. The standard deviation and maximum value can be indicative of whether the average is representative of the landscape's gradient (low standard deviation (*stdev*) and *max* is close to the average) or if there are outlier gradients (high *stdev* or *max* is significantly higher than the average). High values of all three can be interpreted as an extremely rugged landscape.

2.4.4 Fitness-Distance Correlation

The fitness distance correlation measure (FDC) [51] calculates the correlation between the difference in distance and fitness scores between points.

$$FDC = \frac{\sum_{i=1}^{n}((f_i - \bar{f})(d_i - \bar{d}))}{\sqrt{\sum_{i=1}^{n}(f_i - \bar{f})^2}\sqrt{\sum_{i=1}^{n}(d_i - \bar{d})^2}} \tag{2.3}$$

A landscape that has a high correlation between movements in space and their resulting fitness scores will generally attract the search in a single and correct direction. Conversely, landscapes with a low correlation between fitness and distance will pull the search in many directions, increasing the chance that the algorithm get stuck in a local optimum. The next section will summarize two commonly used benchmark generators for dynamic optimization testing.

2.5 Moving Peaks Benchmark

The Moving Peaks Benchmark (MPB) [5, 6, 7] was introduced to provide a benchmark generator which yielded continuous, dynamic, multi-modal optimization functions. Its main focus is to provide a framework under which a number of local optima, also known as peaks, can be controlled and varied over time. More specifically, the generator inputs control the peaks height, width, position in landscape, as well as the severity and frequency of the environment changes. The original MPB [6] function is described as:

$$F(\vec{x}, t) = \max_{i=1}^{n} \frac{H_i(t)}{1 + W_i(t)\sum_{j=1}^{d}(x_j(t) - X_{ij}(t))^2} \tag{2.4}$$

where n is the number of peaks, d is the number of dimensions, t is the current time step value, while H, W, and X are the height, width, and position of the i^{th} peak respectively. The time step value is simply the number of changes that the environment has undergone, i.e., $F(\vec{x}, 2)$ yields the $\vec{x}$ position in a resulting landscape from two changes applied to the original, generated landscape. A change is applied at the given interval which is usually measured in either algorithm iterations or individual function evaluations; the implications of both are discussed later in this chapter. This interval controls the temporal severity of the dynamic environment where the shorter

the interval, the less time (function evaluations) an algorithm has to locate optima before it is changed and moved. The change function works by varying the height, width and position of peaks (H, W, X) and is described as:

$$H_i(t) = H_i(t-1) + h_{severity} * N(0,1) \tag{2.5}$$

$$W_i(t) = W_i(t-1) + w_{severity} * N(0,1) \tag{2.6}$$

$$\vec{X}_i(t) = \vec{X}_i(t-1) + \vec{v}_i(t) \tag{2.7}$$

where $N(a,b)$ is a normally distributed random number with mean a and variation b, $h_{severity}$ and $w_{severity}$ are the height and width severity parameters respectively, and $\vec{v}_i(t)$ is a generated shift vector that is applied to its corresponding peak in order to move the peak's position in the d-dimensional space. The shift vector is obtained by:

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)) \tag{2.8}$$

where $\vec{v}_i(t-1)$ is altered by the addition of a random vector $\vec{r}$ and normalized to the shift length parameter. The shift correlation parameter λ is also applied in order to correlate the current movement with the previous movement to the degree of λ. Figure 2.1 shows a few examples of 2-dimensional MPB environments.

In an attempt to add more variability in terms of change characteristics, Li and Yang [19] made modifications to the original MPB. These modifications include the variance of the number of peaks over time as well as the control over the proportion of peaks affected by a change. The goal of changing peak counts was to analyze the performance and adaptability of algorithms that dynamically vary their population size in order to track a changing number of peaks. The purpose of controlling the number of peaks re-positioned by a change was to study the effects it had on algorithms that relied upon change detection.

2.6 Generalized Dynamic Benchmark Generator

In an attempt to formalize a uniform definition for dynamic environments across the binary, combinatorial, and real optimization spaces, Li and Yang [24] proposed the Generalized Dynamic Benchmark Generator (GDBG). This framework Generalizes a

Figure 2.1: Moving Peak Benchmark: two-dimensional 5-peak, 20-peak, and 100-peak landscapes (varied peak widths).

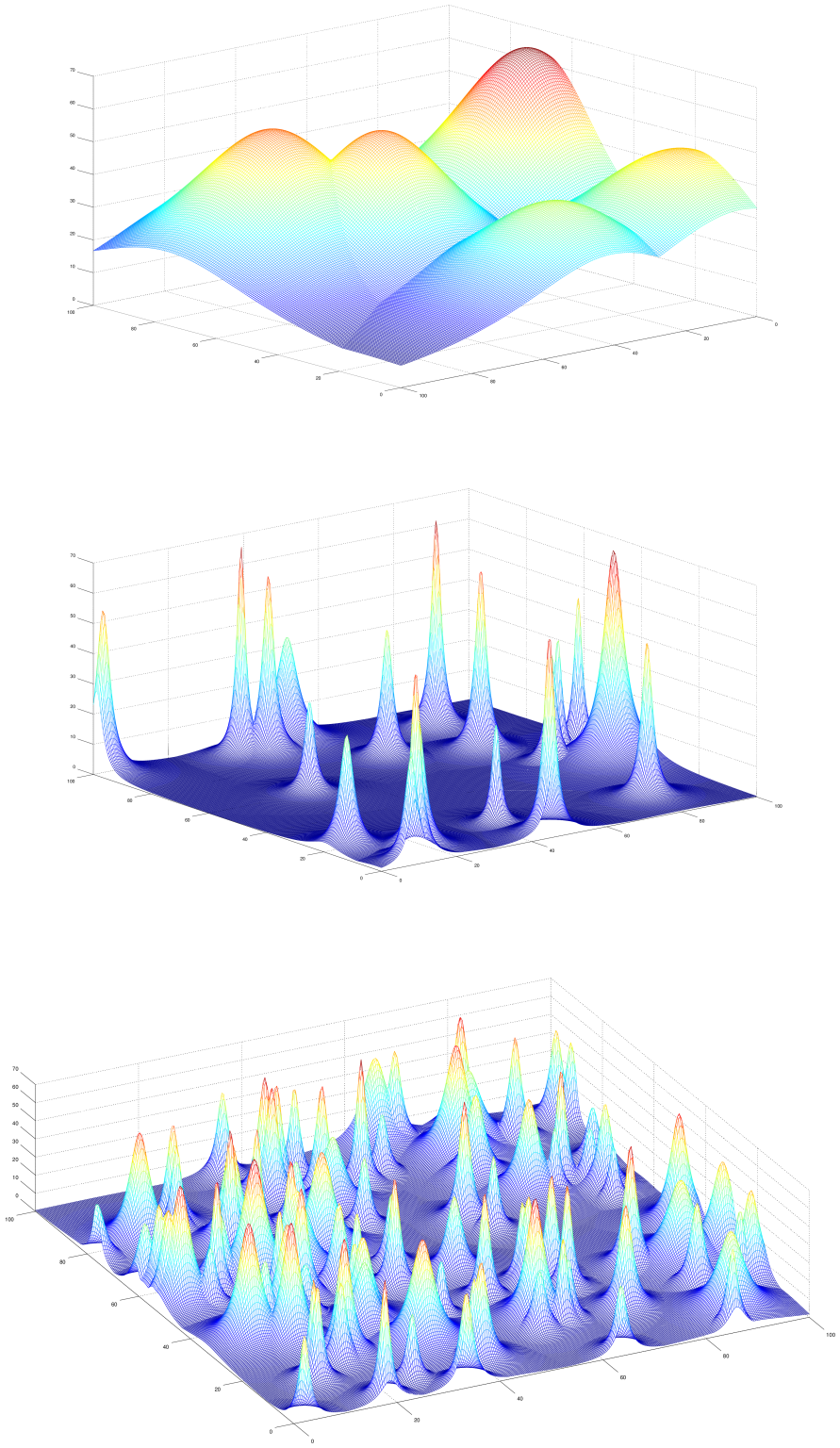


Table 2.1: Original and modified MPB Parameters: typical values found in current literature

Parameter	Symbol	Typical Value Range
	Original Paramaters	
number of peaks	$peaks$	[1, 200]
change frequency	U	[5,000, 25,000] evals
min/max width	w_{min}, w_{max}	[0.1, 12.0]
min/max height	h_{min}, h_{max}	[30.0, 70.0]
width severity	$w_{sevirity}$	[0.1, 1.0]
height severity	$h_{sevirity}$	[5.0, 10.0]
shift length	s	[1.0, 10.0]
shift correlation	λ	[0.0, 1.0]
	Added Parameters	
peak change %	$cpeaks$	[0.1, 1.0]
min/max peaks	p_{min}, p_{max}	[1, 200]
peak Count severity	$p_{severity}$	[5, 90]

dynamic function F as:

$$F = f(\vec{x}, \theta, t) \tag{2.9}$$

The function described has three general inputs; a potential solution $\vec{x}$, the system control parameters θ, and the environment time t. In relation to the previously described MPB function, we can already see that a potential solution x is represented by a d-dimension position in the landscape and time t is represented by the number of changes that occurred. The control parameters θ are those variables that control the fitness distribution amongst the solution space. For the MPB these would be the peak heights, widths, and positions, as well as the bounds and severity variables. A change within the θ is how a change in the solution space is obtained:

$$F(x, \theta, t+1) = f(x, \theta(t) \oplus \triangle\theta, t) \tag{2.10}$$

The $\triangle\theta$ in Equation 2.10 describes a deviation in theta which is categorized in this framework by the following change types.

$$smallStep : \triangle\theta = \alpha * ||\theta|| * r * \theta_{severity} \tag{2.11}$$

$$largeStep : \triangle\theta = ||\theta|| * (\alpha * sign(r) + (\alpha_{max} - \alpha) * r) * \theta_{severity} \tag{2.12}$$

$$randomStep : \triangle\theta = N(0,1) * \theta_{severity} \tag{2.13}$$

$$chaoticChange : \theta(t+1) = A * (\theta(t) - \theta_{min}) * (\theta(t) - \theta_{min}/||\theta||) \tag{2.14}$$

$$recurrentChange : \theta(t+1) = \theta_{min} + ||\theta||(sin(\frac{2\pi}{P}t + \phi) + 1)/2 \tag{2.15}$$

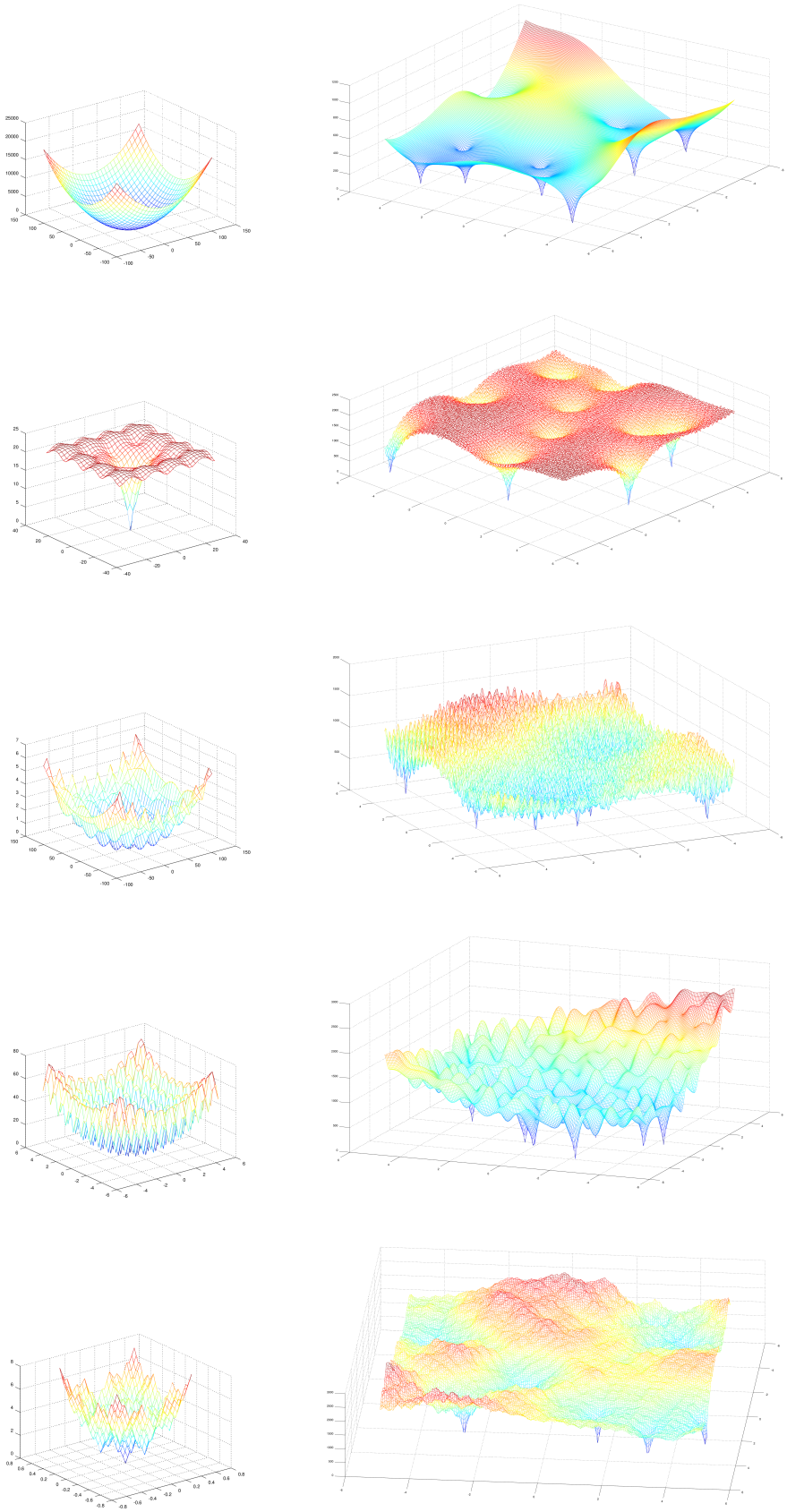
$$recurrentChange : \theta(t+1) = \theta_{min} + ||\theta||(sin(\frac{2\pi}{P}t + \phi) + 1)/2 + N(0,1) * noise_{severity} \tag{2.16}$$

Table 2.2: GDBG non-dimensional change variables

Symbol	variable	default value
α	constant (0,1)	0.04
α_{max}	constant (0,1)	0.1
θ_{min}	min change	-
$\theta_{severity}$	change severity	-
$\|\|\theta\|\|$	change range	-
$N(0,1)$	normal dist. rand. num. mean=0 var=1	function
r	random number (-1,1)	function
$sign(x)$	$(x > 0)?1 : (x < 0)? - 1 : 0$	function
A	constant (1.0,4.0)	3.67
P	period length of recurrent change	-
ϕ	initial phase of recurrent change	-
$noise_{severity}$	noise in recurrent change (0.0,1.0)	-

Table 2.2 lists the different variables listed in the change equations. It should be noted that theses changes are referred to as non-dimensional changes. Dimensional changes are also included in the framework and cause the addition or removal of variables to the problem, therefore changing the dimensionality of the solution representation. Dimensional changes are not considered as they are outside of the scope of this thesis.

Figure 2.2: Dynamic Composition Functions: two-dimensional basic functions and their 10-peak composition (in order): Sphere, Ackley, Griewank, Rastrigin, and Weierstrass



The change types outlined above define a common behaviour for the change function in a dynamic of any solution space type. There are two instances of generators defined under GDBG for real-space optimization, namely the dynamic rotation peak benchmark generator, and the dynamic composition benchmark generator. For the remainder of this chapter, these are referred to as the rotation generator and the composition generator respectively.

2.6.1 Dynamic Rotation Peak Benchmark Generator

The rotation generator [24] is identical to the MPB except for the generalization of control parameters and also in how the peaks positions are altered. The authors in [24] identified that, in the MPB peak shifting mechanism there is a bounce-back effect that causes unequal change. The argument is that when a peak bounces off of a dimensional boundary that its net movement in that dimension is $shift - |pos(t-1) - dist_{toBoundary}|$ which is less than $shift$ itself. Their solution, derived from [27], is to rotate the projection of a peak to avoid the need for boundary constraints on peak movement. To do this, a rotation matrix M is created by randomly coupling the dimensions' indices which make up a plane and assigning a randomly generated angle for each plane, i.e., pair of dimensions (see [25] and [27] for the detailed algorithm). The peak positions are therefore changed by applying the rotation matrix to its peak:

$$\vec{peak}_i(t+1) = \vec{peak}_i(t) \cdot M \tag{2.17}$$

As previously mentioned, the system control parameters are $\theta = (\vec{H}, \vec{W}, \vec{X})$ which are the peaks' heights, widths and positions. Each of these can be assigned one of the non-dimensional change types outlined above. For example, changes to $\vec{H}$ can be defined as $\vec{H}(t+1) = smallStep(\vec{H}(t))$, where $||\theta||$ and $||\theta_{severity}||$ can be set to 40.0 and 7.0 respectively which would yield the typical height settings for MPB described in Table 2.1. Since the positions are altered by the rotation matrix, the typical $||\theta||$ for $\vec{X}$ should be $(-\pi, \pi)$.

2.6.2 Dynamic Composition Benchmark Generator

The composition generator [24] was created to address another issue with the MPB, namely the symmetry of its optima. This is an important issue to address since it is likely that there exists many real-world problems where symmetrical optima are not the case, making the MPB unrepresentative as a dynamic benchmark. The

composition generator is based on the static composition functions proposed in [26] where well known, static functions are combined and rotated in order to produce more challenging landscapes. The rotation is performed to move the optima, located at $x = 0$ for the functions used, in order to identify algorithms with positional favouritism (such as the middle of the landscape). The rotation in the composition generator is used as the spatial operator to move the peaks around the search space.

The functions used in the composition are referred to as basic functions in this framework. The composition is such that for every desired peak in the landscape, one more basic function is added to the composition. The composition is created by stretching the basic functions to fit the desired domain and adding them together as described by:

$$F(x, \theta, t) = \sum_{i=1}^{m} (w_i \cdot (f'_i(\frac{x - \vec{O}_i(t) + O_{iold}}{\lambda_i} \cdot \vec{M}_i) + \vec{H}_i(t)) \tag{2.18}$$

Table 2.3: GDBG Composition function variables' descriptions and defaults

Symbol * bf = basic function	variable	default value
w_i	weight factor i^{th} bf	-
λ_i	stretch factor i^{th} bf	-
σ	coverage factor i^{th} bf	1
f'_i	height adjusted i^{th} bf	function
f_{max}	estimated max value i^{th} bf	function
C	height adjustment constant	2000
$\vec{O}_i$	peak position i^{th} bf	-
O_{iold}	original optima position i^{th} bf	0 *
$\vec{M}_i$	orthogonal matrix i^{th} bf	-

Table 2.3 lists the names and meanings of each of the variables composition generator. The dynamics of the system are controlled like the ones in the rotation generator where $\vec{H}$ and $\vec{O}$ are subject to one of the non-dimensional change types. $\vec{O}_i$ is rotated by its corresponding M_i rotation matrix, constructed in the same way as in the rotation generator. Figure 2.2 illustrates the basic functions and their ten-peak composition landscapes. It can be observed that the landscapes produced are more complex than the MPB.

This chapter provided a survey of landscape features and analysis techniques, along with a summary of current dynamic benchmark generators. The next chapter describes the experiments performed to classify these generated landscapes according to the landscape metrics reviewed in the previous chapter. The landscape metrics described above will be used to categorize the function generator configurations according to the characteristics they produce. The change operator of the environments will also be evaluated to verify whether or not they can produce significant changes to these characteristics.

Chapter 3

Dynamic Landscape Characterization

This chapter presents the experimentation conducted in the classification of dynamic, multi-modal benchmark generators, in terms of their landscape characteristics. The metrics described in the previous chapter will be used in order to quantify the characteristics. Three distinct experiments are outlined in the experimental setup section and summarized in Table 3.1. The purpose of these experiments are to quantify the variability of the metrics used, classify the dynamic benchmark functions, and finally, determine whether the change operators of the benchmark generators have an effect on landscape characteristics.

3.1 Benchmark Characterization

A visual inspection of the different landscapes in Figures 2.1 and 2.2 suggests a fairly comprehensive representation of different landscape features. The following experiments are focused on measuring the characteristics of the static landscapes of these dynamic functions. The results will serve as a means of classification in terms of the characteristics, which in turn can be used to evaluate algorithm performance under the established classifications.

It is assumed here that the dynamics of the MPB and GDBG frameworks are not sufficient to cause any significant change to the underlying characteristics. As mentioned in Chapter 2, this assumption stems from the lack of consideration of landscape features in current dynamic research. Although it is possible with severe changes to peak height that significant variance might be produced in gradient characteristics, it

is believed that the simple shift/rotation and height variance of optima is not related to characteristics such as ruggedness, neutrality, searchability and others. The experiments in this chapter also look at the effects that the dynamics have on the range of characteristics produced by the different landscapes. If these characteristics cannot be altered under the current dynamics then additional operators might be required in order to better represent potential real-world problems that exhibit such dynamic properties.

3.1.1 Experimental Setup

In order to measure the landscape characteristics, the FEM, DM, FDC, and GM metrics from the previous chapter will be applied to the different generated landscapes. Since these metrics are sample-based methods, either by distribution or random-walk sampling, it is necessary to first evaluate the typical variance in readings produced purely by the sampling. To do this, two runs consisting of 30 samples of each metric are applied to the same generated, static landscape. If the metrics are stable then the two runs should be statistically the same. The functions tested are the MPB and the six composition functions under the GDBG framework. Note that the dynamic rotation generator will not be included this analysis as its landscape characteristics are identical to those of the MPB and its dynamics are identical to those of the composition generator. Since these two are included in experimentation, the rotation generator is omitted as redundant.

Given a stable outcome in the metric testing, a second set of experiments is conducted to determine the average and range of each characteristic that each of the functions produce. This is done by applying thirty runs of each metric, each run on a separately generated, static landscape using the same generator parameters. Again, the seven functions mentioned in the stability tests will be tested under various peak counts and dimensionality. These tests will provide a categorization of each of the functions in terms of the landscape characteristics and will also be a baseline for testing the dynamics of the two generators in the third experiment.

The third set of experiments is to evaluate the effects of the generator dynamics on the landscape characteristics producible by each of the functions. To test these effects, thirty dynamic environments are generated, each with ten changes. The metrics are run thirty times per change/landscape as in the second test. The landscape characteristics are obviously not affected by temporal dynamics, and since we are

only interested in whether or not the dynamics could cause a shift in characteristics, only high-severity spatial dynamics are considered since they are the type most likely to cause such shifts.

Table 3.1: Experimental Summary

Experiment 1	Evaluate inherent variance of landscape metrics
Experiment 2	Classify benchmark functions using landscape metrics
Experiment 3	Evaluate variance of landscape features produced by benchmark change operators

Each of the three experiments will be done with the nine total combinations of number of peaks (5, 10, and 50) and dimensionality (5, 10, and 30). These figures were taken from the common configurations used to test various algorithms in the literature. This will also provide insight into the role that modality and dimensionality play in terms of landscape characteristics. Table 3.2 lists settings for each of the metrics.

Table 3.2: Landscape Characteristic measurements setup

Metric	Sampling	Settings
* $sampleSize = 200 * d$,	$d = dimensions$, $rng = domainRange$	
FEM_{micro}	progressive random walk	1% $stepSize$
FEM_{macro}	progressive random walk	10% $stepSize$
DM	Uniform	10% $threshold$
$GM_{avg,std}$	manhattan prog. random walk	$stepSize = rng * d/sampleSize$
FDC	Uniform	-

3.1.2 Results and Analysis

Table 3.5 shows results of the stability tests. The results show that the average standard deviation in both runs are almost identical and that the difference in averages between the two runs always falls inside the average standard deviation of the two runs. These results show low variance between samples meaning the sample sizes chosen for the metrics yield stable measurements.

Table 3.3 illustrates the average difference of the two runs expressed as a percentage of one standard deviation calculated as $\frac{|avg1-avg2|}{(stdev1+stdev2)/2}$. Overall the results show that the average difference of the two runs for all metrics falls within 34 percent of one standard deviation, meaning all metrics tested yield consistent readings. This figure

seems to be skewed however, by the differences in the gradient measures. This is observed by analyzing the individual averages of the metrics. Here we can see that all of the metrics averaged only around 20 percent of a standard deviation while the the gradient measures averaged about 43 percent. This may be caused by the sampling method as it is the only metric to use the Manhattan progressive random walk and suggests that it may be a more volatile measurement than the other measures. Alternatively the increased variance could be caused by the gradient measure's output, which is not normalized to a range like the others. Unbounded output could produce significant outliers, causing greater volatility. As for the individual averages in terms of the functions, there does not seem to be any significant outliers as they roughly fall in the 20-30 percent range.

The $FEM_{micro,macro}$ measures show a 1-3 percent deviation of its (0,1) normalized output range. FDC shows a 1-5 percent deviation of its (-1,1) normalized output range. Although GM measurement deviations are higher, they remain within an acceptable range as their averages from the two runs would not be considered different from a landscape characteristic view point. A Mann-Whitney-U test was performed on the two averages for each measure and no significant difference was found at the 95 percent confidence level. Overall the conclusion reached is that the metrics are stable and that a very small deviation is caused by the sampling nature of the measurements.

Table 3.3: Average difference for each metric expressed as a percentage of 1 standard deviation

	MPB	ackley	griewank	rastrigin	sphere	weierstrass	Avg
FEMmicro	0.3885145715	0.0548557155	0.6807226846	0.2155047464	0.3330511834	0.5803516346	0.3755000893
FEMmacro	0.1700618547	0.1632611431	0.1867635052	0.2657016891	0.1782478695	0.4963057534	0.2433903025
DM	0.2189551143	0.1799684081	0.2593199804	0.1566570036	0.2077208352	0.1391218038	0.1936238576
GMavg	0.6072515331	0.2587713658	0.5743413383	0.2663652448	0.7363612631	0.6006800499	0.5072951325
GMstd	0.6448765877	0.2621714354	0.5694077704	0.2864351261	0.8785397344	0.6269003502	0.544721834
FDC	0.3870860002	0.0474579826	0.2884849233	0.1615394915	0.2581407804	0.0741465879	0.2028092943
Avg	0.4027909436	0.1610810084	0.4265067004	0.2253672169	0.4320102777	0.4195843633	0.3445567517

Appendix A.1 lists the results obtained by the function characterization tests described in the previous section. As seen in the first set of experiments, the deviations for each of the metrics are quite low despite the runs coming from different static landscapes of the same generator setup. This suggests that the function generators are very regular in terms of the characteristics they initially generate.

Table 3.4: Average difference in deviation for each metric expressed as a percentage of 1 standard deviation

	MPB	ackley	griewank	rastrigin	sphere	weierstrass	Avg
FEMmicro	0.1071976209	0.1385285494	0.1109641871	0.2745650334	0.2170798394	0.0354473999	0.147297105
FEMmacro	0.0698123219	0.198769739	0.2264836202	0.1118013216	0.1561209536	0.1237945377	0.1477970823
DM	0.1583607827	0.1722276242	0.126459562	0.1178014352	0.2643202379	0.1938701452	0.1721732979
GMavg	0.4945410045	0.2562042531	0.5524290906	0.2657163256	0.6239337368	0.3298716647	0.4204493459
GMstd	0.7413346989	0.2497994548	0.5710534952	0.2651555057	0.6011997154	0.330868305	0.4599018625
FDC	0.0926236166	0.0917766438	0.1270560833	0.222617872	0.2017963209	0.2315678227	0.1612397265
Avg	0.2773116743	0.1845510441	0.2857410064	0.2096095822	0.344075134	0.2075699792	0.2514764034

The MPB shows similar readings for ruggedness at the micro and macro level and FEM_{micro} shows lower values across all MPB tested environments. This is to be expected since its peak function is smooth and perfectly correlated to the distance from the peak. The higher values at the macro scale is simply due to the higher odds of stepping from one peak to another with an increased step size of the random walk, 10 percent (macro) of function bounds versus 1 percent (micro). This is supported by the slight increase in both micro and macro readings as the number of peaks are increased as well. The number of dimensions does not seem to have an affect on macro ruggedness as it remains unchanged as the number of dimensions increases. There is, however, a diminishing effect to the micro ruggedness as it decreases with the increase of dimensionality in every peak count. This may also be due to the nature of the random walk; as you increase the number of dimension the odds of stepping from one peak to another diminishes because steps are only taken in one dimension. Gradient readings were very stable as the GM_{avg} and GM_{std} were relatively unchanged in all nine MPB environments. The two readings were very low and also virtually identical, indicating no steep gradients. This is also expected as mentioned earlier, the fitness is perfectly correlated with distance to the nearest, highest peak. The FDC readings are the highest for the MPB for this same reason; it does not measure an FDC reading of 1.0 (highest searchability) only because the function is multi-modal. This argument is supported by the decrease in FDC readings in all three cases that the number of peaks were increased.

The ackley composition ranks among the highest for both micro and macro ruggedness with readings in the mid 0.80 range. Unlike the MPB, this function has many smaller local optima which dominates most of the landscape. This can explain why the increase in peaks does nothing for micro ruggedness while providing a small decrease at the macro level, but this decrease diminishes as the dimensionality increases; once again due to the dominance of the plateau of small local optima which can be

Table 3.5: Average metric readings of two separate samples taken from a single generated landscape to show the stability of the metrics

Domain	R(0.0:100.0)Î10 (MPB)		R(-5:5)Î10 (compositions)		
		Avg 1	STDev 1	Avg 2	STDev 2
MPB	FEMmicro	0.2647055193	0.0307946478	0.2787744885	0.0289382989
	FEMmacro	0.3150104097	0.0077149353	0.313857243	0.0084393189
	DM	-0.1992785495	0.0147871109	-0.1931674095	0.0117411046
	GMavg	6.2428001983	1.685568081	6.9689800201	3.6985623148
	GMstd	5.4857095077	0.7350472536	5.6251084209	2.4200420413
	FDC	0.4751083577	0.0953076092	0.4402795684	0.0783514098
ackley	FEMmicro	0.8689330395	0.0046176168	0.8685209868	0.0041982692
	FEMmacro	0.8478085969	0.0132208405	0.8478738192	0.010880177
	DM	0.1229205145	0.0126678436	0.1223109668	0.0093558375
	GMavg	109.2566274512	20.37846997	99.4146516847	18.0498129937
	GMstd	81.1584057678	15.0144158366	73.2074645611	13.0634809492
	FDC	0.0718042498	0.0577589111	0.0711594903	0.0600393276
griewank	FEMmicro	0.5849765204	0.070561663	0.5562390302	0.070997383
	FEMmacro	0.6463007976	0.0168245838	0.6397579842	0.0200418016
	DM	0.1317569957	0.0132787234	0.1349286657	0.0148042194
	GMavg	10.5070637115	5.0872119797	7.7896710709	3.7911734769
	GMstd	6.9591091614	3.5844463543	5.0058196704	2.355238819
	FDC	-0.1175911509	0.09663118	-0.1711852742	0.0910222698
rastrigin	FEMmicro	0.6035261347	0.0133787369	0.6011489651	0.0160871954
	FEMmacro	0.8323831579	0.0114054948	0.8305154027	0.0098478877
	DM	0.1778011606	0.0148856455	0.1741467668	0.01323487
	GMavg	14.8112543481	3.1948520071	14.3642912988	4.5915977825
	GMstd	10.8747117073	2.2552394951	10.5510507911	3.3700310389
	FDC	0.0122209054	0.0695904605	0.0059071224	0.1091123214
sphere	FEMmicro	0.5737728404	0.0761330668	0.5264542674	0.0507279952
	FEMmacro	0.6436608101	0.0206177729	0.6403113712	0.0145498135
	DM	0.0817279278	0.0144629983	0.0819425531	0.0178759425
	GMavg	7.6266623965	3.4953703852	4.4830485016	1.1262190636
	GMstd	5.2925890704	2.5907215086	2.7025189394	0.736526559
	FDC	-0.3015921042	0.0936230353	-0.2836855256	0.1099289435
weierstrass	FEMmicro	0.7892950565	0.0090059637	0.7838858355	0.009069087
	FEMmacro	0.8076787406	0.0060016765	0.8099325713	0.0066924643
	DM	-0.0910479067	0.0109211941	-0.0922786384	0.0125497464
	GMavg	27.8428368194	4.7791869533	24.6490789198	6.1521000033
	GMstd	20.9103512035	3.5587075247	18.3773674073	4.4610563348
	FDC	-0.3079565677	0.0819594452	-0.3073021264	0.0758119127

observed in Figure 2.2. The sharp narrow peaks also explain the extremely high gradient values which tend to increase with dimensionality. The decrease of gradient values with the increase the number of peaks is once again do to the higher odds of stepping from one peak to another with more peaks, thus cutting the steepness of the step. The GM_{std} is also very high reinforcing the FEM readings in support of a highly rugged landscape. Despite its ruggedness, the ackley FDC readings are comparable to the other composition functions.

The griewank composition demonstrates mid-range macro ruggedness (mid 0.60) across all nine environments. However, micro ruggedness ranges from 0.33 to 0.79.

The micro readings decrease with increases in either modality or dimensionality, however the decrease caused by dimensionality is far greater. Looking at the graph in Figure 2.2 this might be cause by the the extremely high number of local optima which increased exponentially with increase in number of dimensions; introducing a near-smoothing effect at the micro scale. This might explain the large jumps in micro ruggedness in terms of dimensions but not in number of peaks. This argument is also supported by a similar decrease in gradient values which decrease as the landscape is smoothed-out. The gradient values themselves a re low in all cases putting it in the same class as the MBP.

The rastrigin composition function exhibits consistent mid-range micro (0.50–0.59) and high macro (0.75–0.84) ruggedness. Similar to griewank's inverse relation of FEM_{micro} to number of peaks, both the micro and macro readings decrease slightly with increases in either modality or dimensionality. This might again be caused by a smoothing effect when the number of local optima increases, but with a much milder effect in this case. The gradient readings are higher than the MPB and griewank but nowhere near ackley.

The macro ruggedness for the spherical function has a consistent reading of around 0.63 where the micro levels are in the mid 0.50 range; except for in 30-dimensions, where it is 0.35 for all peak counts. As expected, the gradients are very small.

Like the ackley function, the weierstrass composition has a consistently high micro and macro ruggedness, but is slightly lower than ackley. These values are not affected by the dimensionality or modality. The gradients show a consistent mid-range reading with a high standard deviation meaning high ruggedness as well. The gradient values show an increase with the increase in dimensionality.

The Fitness-Distance Correlation (FDC) shows an overall trend across all tested functions of an increase when the dimensionality increases. This might be caused by the nature of its calculation: when the number of dimensions are increased, the total distance between all samples increases. As this distance grows, the correlation between it and the fitness approaches 1 since the range of fitness values are insignificant in comparison to that of distance. One way to address this is with a sample size that is a multiple of the number of dimensions. However, such a sample size would only grow linearly while the search space grows exponentially. For this reason the

FDC should only be used to compare the functions relative to each other. In this regard, all of the composition functions are fairly similar with the exception of the weierstrass composition which is significantly lower than the others. The MPB is the only function with a consistent reading well above 0; reasons for this are explained above.

Appendix A.2 shows the third set of experiments where the metrics were evaluated on environments in which ten spatially-severe changes occur. In comparison to the results in Table A.1 many of the same trends can be identified:

- The average FEM_{micro} value fell within 1 standard deviation of the static readings.
- The average FEM_{macro} value fell within 0.23 standard deviations of the static readings.
- The standard deviations did not rise significantly in terms of any of the measures/environments.
- FDC shows same relative trends as described above.

As expected the spatial dynamics did affect the gradient readings. From an overall perspective, the changes lowered the gradient readings in those landscapes with higher readings, namely the ackley and weierstrass compositions, and increase the rest of the functions that had lower gradient values. It should be noted however that the differences between the two GM_{avg} in all cases fell within the range of GM_{std}. Overall, the benchmarks were categorized according to Table 3.8 regardless of whether the change operators were applied.

Table 3.6: Average difference between static and dynamic runs for each metric expressed as a percentage of 1 standard deviation

	MPB	ackley	griewank	rastrigin	sphere	weierstrass	Avg
FEMmicro	1.2299498848	0.132658652	1.6424474568	1.1172182101	1.7392965569	0.127638024	0.9982014641
FEMmacro	0.2493366604	0.1424441665	0.2378165978	0.1799778634	0.2271851091	0.3572842095	0.2323407678
DM	0.5815860405	0.2141867213	0.1740531433	0.1715530202	0.1434557362	1.0475360194	0.3887284468
GMavg	0.7760957614	1.7043565703	1.6064155086	1.4518346371	1.7001690964	0.9516673677	1.3650898236
GMstd	0.6850907457	1.6743500203	1.5509260189	1.440315585	1.641849427	0.9679961245	1.3267546536
FDC	0.2611652094	0.1886215693	0.1866613977	0.1976741643	0.1812295784	0.5337731628	0.2581875137
Avg	0.6305373837	0.67610295	0.8997200205	0.7597622467	0.9388642507	0.664315818	0.7615504449

Table 3.7: Average difference in deviation between static and dynamic runs for each metric expressed as a percentage of 1 standard deviation

	MPB	ackley	griewank	rastrigin	sphere	weierstrass	Avg
FEMmicro	0.5042787447	0.2567569942	0.7957636352	0.3273261675	0.8310564214	0.2248777879	0.4900099585
FEMmacro	0.4309989192	0.1953241907	0.2679078498	0.1834733833	0.2406758312	0.2176923452	0.2560120866
DM	0.1532239689	0.2453329588	0.1882616949	0.1751185519	0.1278643947	0.4948737105	0.2307792133
GMavg	0.655414898	0.795936728	1.0419134497	0.8364584558	1.1536116603	0.4650736665	0.8247348097
GMstd	0.6182552406	0.7832943544	1.0032551449	0.8215806074	1.1209276496	0.4684739162	0.8026311522
FDC	0.1829270468	0.0920269423	0.1163447724	0.1501055444	0.1247545273	0.2195858552	0.1476241147
Avg	0.4241831364	0.3947786947	0.5689077578	0.4156771184	0.5998150807	0.3484295469	0.4586318892

3.1.3 Conclusion and Characterization

Table 3.8: Generator function landscape characteristics classification; *Griewank FEM_{micro} decreases significantly with dimensional increase

Characteristic	Low	Med	High
Micro Ruggedness (1%$step$)	MPB Griewank$*^{>50}$	Sphere Rastrigin Griewank$*^{10-50}$	Ackley Wierstrass Griewank$**^{1-10}$
Macro Ruggedness (10%$step$)	MPB	Griewank Sphere	Ackley Wierstrass Rastrigin
Gradients	MPB Sphere Griewank	Wierstrass Rastrigin	Ackley
Searchability	MPB	Ackley Rastrigin Griewank Sphere	Wierstrass

In conclusion, the six functions analyzed in this study cover a fairly wide range of the landscape characteristics covered by the presented metrics. The general classifications by characteristics were made based on the observations of experimental results in Appendix Tables A.1 and A.2 and is described in Table 3.8.

It has been shown in these experiments that despite a slight increase in the mean deviations of the average metric readings, the dynamics of either system did not significantly change the characteristics of any landscape tested. The only significant variance to the characteristics within a given function came from the modality and dimensional settings. This suggests that the dynamics of both systems are missing control parameters that would allow these and other landscape characteristics to be a part of the spacial dynamics of the environment.

Chapter 4

Particle Swarm Optimization

This chapter provides the fundamentals of the particle swarm optimization (PSO) algorithm since it is the basis for all of the multi-modal variants being compared in Chapter 6.

4.1 Particle Swarm Optimization

PSO is an optimization algorithm developed by Eberhart and Kennedy [9] which was inspired by the behaviour of bird flocks. The algorithm can be described from a high-level as a group of particles flying around an n-dimensional search space, communicating individual findings within the swarm. The individuals posses a memory construct which is used to reference the best position and fitness score of past exploits. As will be discussed below, a particle's movement through the search space is partially based on the position of the best solution it has found so far and thus a particle must use its memory to keep track of that position.

Each PSO particle position is composed of values to the problem's variables, representing a potential solution to the problem. The particles' positions are changed iteratively based on its velocity which is influenced by the best positions found, both by the particle in question as well as the swarm it belongs to. The idea behind this behaviour is to attract particles to the best regions found so far by the swarm while they oscillate through the search space. The velocity and position update equations are shown in Equations 4.1 and 4.2, respectively.

$$v_i^d(t+1) = w * v_i^d(t) + r_1 c_1 (x_{pb_i}^d - x_i^d) + r_2 c_2 (x_{gb}^d - x_i^d) \tag{4.1}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{4.2}$$

The acceleration coefficients, denoted by c_1 and c_2, determine the level of attraction to the personal best ($\vec{x}_{pb_i}$) and swarm best ($\vec{x}_{gb}$), respectively, and are also known as *cognitive* and *social* influence factors. These coefficients generally control the speed of convergence; if cognitive influence is larger than the social, the swarm will converge slowly, if at all, since particles are pulled strongly toward their own personal best positions instead of the swarm's best. Conversely, a larger social influence will cause fast and maybe premature convergence. The random vectors $\vec{r_1}$ and $\vec{r_2}$ are generated and applied to the cognitive and social influences respectively in order to give the particles a stochastic behaviour. The optimal values for the acceleration coefficients are problem dependant.

The inertia weight w in the velocity calculation deals with the concept of momentum by dictating how much of the previous velocity will be present in the updated value. Higher values of inertia make the velocity more resistant to sudden change in terms of direction since such updates will have to fight the momentum of the previous direction. Conversely, smaller values encourage faster convergence since the velocity will have little or no momentum which encourages quick positional alignment with the attraction areas.

PSO starts the search by initializing the particles' positions and velocity vectors to $\vec{v} = 0$. This process should distribute the particles throughout the entire search space which is important for exploration. At each iteration of the algorithm, the velocity is updated according to Equation 4.1 and then it is applied to the particle's position in Equation 4.2 in order to relocate the particle. Once moved, the fitness is calculated based on the new position which is stored in the particle's memory if the new position is better, in terms of fitness, than the current memory position. This personal memory is often referred to as personal best (*pbest*). The swarm also remembers the best position that any particle has encountered. This means that at every evaluation of a particles position the swarm's best is updated in the same fashion as the *pbest*. The swarm's memory is often referred to as the global best (*gbest*). Algorithm 1 [9] shows the original PSO procedure.

Algorithm 1 Original PSO Algorithm

```
for all Particles p do
  initialize p.position
  p.bestPosition ← p.position
  p.bestFitness ← f(p)
  if p.bestFitness is better than gBest.bestFitness then
    gbest ← p
  end if
end for
while stopping condition not met do
  for all Particles p do
    update p.velocity using eq. 4.1
    update p.position using eq. 4.2
    if f(p) is better than p.bestFitness then
      p.bestPosition ← p.position
      p.bestFitness ← f(p)
    end if
    if p.bestFitness is better than gBest.bestFitness then
      gbest ← p
    end if
  end for
end while
```

4.2 PSO in Dynamic Environments

This section outlines the challenges encountered in applying the PSO algorithm to dynamic problems.

4.2.1 Environment Change and Memory Systems

The PSO algorithm was originally created for static environments and thus requires two main issues be addressed in the adaptation for dynamic contexts. The first issue is one of outdated memory and is a common problem to algorithms that use state-memory systems like the PSO particle's best seen position. The problem with memory in a dynamic landscape is that a change in the fitness landscape invalidates the memory because the associated fitness value for the stored position has potentially changed. This could lead to false attractors within the swarm; *gbest* positions whose fitness value is inferior to others however is interpreted as being superior due to the outdated, overrated fitness value stored.

One way of dealing with the issue of outdated memory is to re-evaluate the memory position at each iteration to ensure that the fitness has not changed [59] and in the event that it has and the current position is better, replace it with the current position. The advantages of this method is its simplicity. The disadvantage is that it doubles the function evaluations required as the stored best is evaluated along with the current position. When change is detectable, the information of when changes occur can be used in substitute of constant re-evaluation of memory, making it only necessary to evaluate when a change is known to have occurred. This yields a more efficient strategy in environments where change is infrequent. Change detection will be discussed in more detail in the context of the next dynamic issue.

4.2.2 Population Diversity and Search Potential

Another common challenge among most population-based algorithms in dynamic environments is their convergent behaviour [1, 59]. Convergence is an important feature that allows the search algorithm to assign more individuals to a promising area in the effort of exploitation. This behaviour, however, becomes a problem in dynamic environments since a converged population loses its ability to explore the broader landscape. More specifically, a converged population is one whose members can all be found in the same neighbourhood of the search, thus creating a lack of positional diversity to attract individuals beyond that neighbourhood. The following two particle behaviours were introduced to address the issue of swarm diversity.

4.2.2.1 Charged Particles

The charged particle (charged PSO) [8] was developed for the purpose of limiting the convergence of a swarm without losing the exploitation benefits of convergence itself. This is done by assigning a new particle behaviour to a portion of the swarm's particles, whose job it is to maintain swarm diversity. The charged behaviour uses a different velocity update equation inspired by Coulomb repulsion [8]. This new update equation, defined by Equation 4.3, has properties which force charged particles away from each other by a factor of their proximity. The Q terms represent the charge value of assigned to the particle and $\vec{r_{ij}}$ is the vector between particles i and j where $\vec{r_{ij}} = \vec{x_i} - \vec{x_j}$.

$$v_i(t+1) = w * v_i(t) + c_1(r_1(\vec{y_i} - \vec{x_i})) + c_2(r_2(\vec{y_i} - \vec{x_i})) + \sum_{i \neq j} \frac{Q_i Q_j}{|\vec{r_{ij}}|^3} \vec{r_{ij}} \tag{4.3}$$

The repulsion inhibits the charged particles from converging with each other. The charged particles role in this system is to orbit the converging swarm and provide the diversity required to attract the swarm to the shifted location of the optima being tracked in the event of a change.

4.2.2.2 Quantum Particles

It is possible that charged particles achieve high velocities due to the unbounded denominator of the charged velocity Equation 4.3 [2, 3]. These high velocities cause rogue particle behaviour where it will diverge far from the search space [2]. To address this issue, as well as the undesirable n^2 computational complexity of Coulomb's repulsion, the quantum behaviour, known as quantum PSO (QSO) [2] was developed. QSO replaces the charged behaviour in the attempt to simplify the diversity maintenance using the quantum update equation given in Equation 4.4.

$$v_i(t+1) = d(r_{cloud}) \tag{4.4}$$

This equation is inspired by the non-deterministic positioning of matter at the quantum scale. A quantum particle moves to different locations of the search space around the converging particles, also referred to as a nucleus swarm, according to a given distribution model d. The distribution of the quantum behaviour is defined by the quantum cloud radius r_{cloud}. This behaviour guarantees that the swarm's search radius remains constant throughout the search.

4.2.3 Change Detection and Alternatives

Two general approaches to the problem of diversity maintenance are change detection and continuous diversity maintenance [2]. Algorithms using change detection treat dynamic environments as a series of discrete static environments. When static PSO starts, the particles are randomized in terms of their position in order to achieve good diversity in the interest of initial exploration. Change detection is a mechanism that tries to recognize when the environment has changed (transitioned into the next static environment) which triggers a reaction to re-diversify the population so that the new landscape can be explored. The advantage of change detection is that it is efficient in management of diversity, only injecting it when necessary. The disadvantage is that some dynamic environments can be hard or even impossible to properly detect change in. These environments can pose challenges such as noise or partial landscape changes. Noise can cause constant triggering of the diversity mechanism

if said mechanism was based on re-evaluating a position in the landscape looking for a change in fitness value. This would inhibit the algorithm from exploitation due to constant re-diversification. Some change detection mechanisms use difference thresholds to filter out noise, but this assumes that the noise in the fitness function is known and regular. Partial-landscape changes might cause change to go undetected if there is no detection agent in the localized region of change. These types of changes can occur if untracked optima change or if new optima appear, and will have a negative impact on performance due to the lack of detection.

Constant diversity maintenance can be described as any strategy that does not use change detection [2]. These methods usually consist of modifying particle behaviour to limit convergence, using a derived metric of population diversity in order to detect convergence itself. The advantage of these strategies is that they are more suitable for environments where changes are hard to detect [16]. The disadvantage is that search performance can be negatively affected by the constant focus on diversity [20], if not properly configured for the given environment.

4.3 PSO in Multi-Modal Landscapes

Multi-modal landscapes are an issue for the standard PSO algorithm because of its convergent behaviour [1]. Convergence itself is a desirable behaviour for population based algorithms as it encourages the fine-grain optimization of a particular area in the search space. A side-effect of convergence where this fine-grain optimization happens before the population has searched the broader landscape resulting in the potential exploitation of a sub-optimal area. This primary exploration phase is not only affected by the social and cognitive coefficients c_1 and c_2, but also the inertia weight w, the initialization distribution, as well as number of particles used.

4.3.1 Local-Best PSO

The original PSO described above is often referred to as global-best or *gbest* PSO. The first multi-modal variant of PSO, called local-best *lbest* PSO, was created [12] in order to address the issue of premature convergence. *lbest* PSO works by separating the population into s neighbourhoods where the social/swarm memory is only shared by particles within the same neighbourhood. This strategy attempts to give the algorithm a better chance at converging on the area containing the global op-

tima since attraction to sub-optimal areas are limited to only n/s particles leaving other swarms free to pursue other promising areas. This separation of particles into neighbourhoods does not guarantee that the neighbourhoods are distinct areas of the search space since there are no defined mechanisms to exclude sub-swarms from each others' neighbourhoods [29].

4.3.2 Constriction PSO

The constriction PSO behaviour [10] is yet another PSO variation which is employed by some of the algorithms reviewed in Chapter 5. This behaviour replaces the inertia weight w for the constriction factor χ. Overall, the modified velocity update is given in Equation 4.5.

$$v_i(t+1) = \chi[r_1 c(x_{pb_i} - x_i) + r_2 c(x_{gb} - x_i)] - (1-\chi)v_i(t) \tag{4.5}$$

Given certain values for c_1 and c_2, the constriction PSO guarantees convergence which is desirable for niching sub-warms that have a priority of exploitation over exploration.

This chapter summarized the PSO algorithm as well as the challenges that must be considered when applying PSO to dynamic problems. Numerous niching techniques and operators have been developed to address the issue of distinct neighbourhoods identified in Section 4.3; these works are reviewed in the next chapter.

Chapter 5

Dynamic Multi-Modal Particle Swarm Review

This chapter summarizes various attempts at identifying and tracking multiple optima in dynamic environments. For each algorithm, their niching parameters are discussed and classified in terms of the behaviours they influence.

5.1 Niching Algorithms

Multiple optima can be located either sequentially or in parallel. Sequential methods involve locating a single optimum and then restarting the search such that previously found optima are avoided. An example of this is the de-rating of a function [58] where the fitness function is altered in order to eliminate attraction to already-identified areas. This contrasts to the parallel methodology, where multiple optima are simultaneously identified. Although both strategies can be employed in a static context, only those of the parallel variety are suited for dynamic environments due to the time constraints imposed before the environment changes. Therefore, the algorithms reviewed in this chapter all use parallel methods of niching.

5.1.1 Speciation PSO

Speciation PSO (SPSO) was introduced by Parrot and Li [15, 16] and uses the Euclidean distance between particles as a basis for separating a main swarm into sub-swarms with the goal of location multiple optima. These sub-swarms are referred to as species.

A particle in the speciation framework is defined as either a species seed or a species member. Being a seed particle $\vec{S}_{seed}$ simply means that the particle's *pbest* position is the best of all the *pbest*'s in a given species. A species' bounds are defined by a hyper-spherical radius r which uses the species seed as the center. Therefore, a particle $\vec{p}$ belongs to a species S if the Euclidean distance between $\vec{p}$ and the $\vec{S_{seed}}$ is less than the radius r. If a particle falls within multiple species then it joins the sub-swarm with the best seed fitness value. Conversely, if a particle does not fall under any species then it forms a species by itself and essentially adopts a cognitive-only behaviour. This segregation of the particles happens at every iteration which means that the species are constantly redefined.

SPSO also attempts to limit the number of particles within a single species in order to encourage the exploration of new potential optima. This is done by defining a $pmax$ value which is used as a maximum sub-swarm size S_{size}. When the number of particles in species S exceeds the given $pmax$ value, the m worst particles are then re-initialized, where $m = S_{size} - pmax$. This convergence-based re-diversification is done without the use of a change detection method.

A pitfall of the SPSO algorithm is the static nature of the inputs r and $pmax$. With a static radius r the algorithm assumes symmetric basins among optima. Although this is assumed in some benchmark functions such as the commonly used moving peaks benchmark (MPB), it is certainly not representative of all real-world landscapes. Setting a static $pmax$ also assumes that an appropriate sub-swarm size is the same for every peak, which again is not likely to be the general case.

5.1.2 Multi-swarm PSO

The multi-swarm framework was developed by Blackwell and Branke [1] to track multiple peaks in dynamic environments, specifically environments where the spatial change severity of peaks is low and the number of peaks is known in advance. The algorithm uses a predefined, constant number of sub-swarms during the search. These sub-swarms use either the charged (mCPSO) or quantum (mQSO) behaviours, described in Chapter 4, in order to maintain diversity. The algorithm can still be used on environments whose peaks are unknown or are not constant during search, although the maximum number of optima the algorithm is able to track is predefined and fixed. This is undesirable in environments where the number of optima is not known in advance or changes over time.

Blackwell and Branke [1] introduced an exclusion mechanism among sub-swarms in order to guarantee that each swarm optimizes a separate area in the search space. Exclusion is an important swarm interaction which should be included in multi-population techniques, in one form or another, in order to separate sub-swarms within the search space. The exclusion operator can be thought of as collision detection for sub-swarms. A collision is described in [1] as two swarms whose swarm attractors are within a predefined distance r_{excl} of each other. If a collision is detected, the sub-swarm with the worst attractor in terms of *pbest*, is re-initialized. A swarm attractor in this context is simply a swarm's *gbest* particle.

Blackwell and Branke [2] introduced another swarm operator in order to deal with the issue of diversity maintenance. The anti-convergence operation checks the convergence status of all sub-swarms and re-initializes the worst performing swarm in the event that they are all deemed converged. A sub-swarm is considered converged if the swarm diameter of the neutral particles is less than a predefined, constant value, parametrized by r_{conv}. The swarm diameter, in this context, refers to the search diameter of the neutral particles and is defined as the greatest distance between any two neutral particles in the swarm. This only applies to the neutral particles as the orbiting particles (charged or quantum) do not converge. This anti-convergent behaviour attempts to ensure that there is always at least one swarm exploring the broader space for new local optima.

5.1.3 Hierarchical Clustering PSO

Clustering PSO (CPSO) was proposed by Li and Yang [20]. The criteria used to divide the sub-swarms in CPSO is similar to SPSO in that the euclidean distance between individuals is used to gather like particles. Like SPSO, and unlike the multi-swarm method in [2], this approach to clustering does not require the number of optima to be known in advance. The clustering method used begins by classifying all of the randomly initialized particles as individual clusters. The algorithm then combines pairs of clusters until every cluster has a minimum population size of two particles. Two clusters are combined when their distance from each are is the smallest among all pairs of clusters. The distance between clusters is defined as the smallest Euclidean distance between two particles, each from one of the clusters. A maximum cluster size is also defined as $subSize$ where two clusters will not be joined if their combined population size exceeds this limit. This method of clustering is given the name single linkage hierarchical clustering [20].

Once the initialized population has been clustered, the iterative search begins and consists of applying the standard PSO's velocity and position update equations to each of the clustered swarms. A series of cluster operations are also applied after each swarm update which consist of exclusion, overcrowding, convergence, and regeneration.

The exclusion operator is similar to the one in the multi-swarm algorithm in [2], however, the collision detection between swarms uses an overlap calculation instead of the distance between the swarm centers. The overlap is calculated based on the proportion of particles from one swarm that are found in the other swarm's radius. A swarm's radius is defined as being the average distance between a particle of the swarm and it's center. If this proportion is larger then a predefined, constant value *overlap* then the swarms are merged. Merged swarms resulting in a population size larger than $subSize$ have the excess particles removed according to their *pbest* values. These excess particles are removed completely from the search space and are not simply re-initialized. This overcrowding operator assures limitations on the resources assigned to any individual swarm.

When swarms converge, their movement is negligible and their function evaluations become redundant. The convergence operator [20] works by taking the radius of a swarm, calculated previously by the exclusion operator, and uses it to determine whether or not a swarm is converged. If the radius is below some predefined value ϵ, then the swarm is removed.

The convergence and overcrowding operators [20] can eventually lead to a depletion of particles. Therefore, a generating operator is applied which restores the main swarm to its initial size, paramaterized by $gSize$. When a change is detected, all sub-swarms are replaced by a newly initialized main swarm and the clustering operation is repeated in order to cluster the new particles.

In the successor variant CPSOR [19], a substitution of the regeneration operator is made for one that does not rely on change detection. This new operator monitors the main swarm size and triggers an injection when that size falls under a predefined proportion of $gSize$. Once triggered, the number of particles required to restore the main swarm size to $gSize$ is created, leaving existing sub-swarms intact. The goal of this variant is to operate in environments where change detection has higher failure

rates, such as noisy environments.

5.2 Adaptive Variants

The algorithms summarized in this section represent adaptations of the three frameworks presented above, and are aimed at eliminating fixed niching parameters.

5.2.1 Adaptive Niching PSO

Adaptive-Niching PSO (ANPSO) was created by Bird and Li [17] in attempt to make the speciation-related parameters, r and $pmax$ from SPSO, adaptable to a given optimization problem. The niching method presented in ANPSO rids itself of a predefined radius used to create niches by making it adaptive. In SPSO [15, 16] the radius is used to define a species radius and, as pointed out above, the shape and size of the basins cannot always be assumed. The radius in ANPSO is used to form connections between particles which is represented in the form of a graph structure, where the particles are nodes and the connection between them are edges. A connection is made between two particles if the distance between them is less than the adaptive radius, which is calculated as the average distance between each particle and its nearest neighbour [17]. Particles are connected when they have been closer than the average nearest neighbours for a minimum number of iterations, parametrized by $iter_{min}$. Connections are also broken if the distance between the connected particles becomes higher than the average nearest neighbour for a given number of consecutive iterations, parametrized by $iter_{length}$.

The niches are formed from the connected sub-graphs meaning two particles, referred to as a and b, are part of the same niche n if there exists a path of any length from particle a to b. Under this niching framework, particles a and b can be part of the same niche even if the distance between them is greater than the radius. This happens through a connected group of particles C, whose distance to particles a and b is smaller than the radius where $|C| >= 1$. This is a less rigid definition of membership and allows for niches to form more easily. The size of these niches are also governed by a $pmax$ value, identical to SPSO where excess particles are re-initialized.

Un-niched particles are those whose nodes have no connected edges. These particles are put into a Von-Neumann topology [14] to encourage the eventual convergence

of particles who remain un-niched for longer periods of time. This convergence happens by sharing information between the un-niched particles so that those in weaker regions will be eventually attracted to others in stronger regions, thus forming niches themselves.

5.2.2 Self-Adaptive Multi-Swarm

An adaptive variant of the multi-swarm algorithm [2] is introduced in [4] as the self-adaptive multi-swarm optimizer (SAMO). The purpose of this was to address the predefined, fixed number of sub-swarms necessary in the original framework. A new parameter n_{excess} is introduced to control the number of free swarms that are allowed to exist at any given time. A free swarm is defined as one whose search diameter, previously discussed in relation to the system's exclusion operator, is greater than the convergence parameter r_{conv}. A larger diameter is interpreted as a swarm searching a broader area and yet exploiting a specific optima. The parameter n_{excess} is used at every iteration to add or remove a swarm when the number of free swarms is below or above n_{excess} respectively. The exclusion and anti-convergence operators remain the same as mCPSO/mQSO.

5.2.3 Adaptive Hierarchical Clustering PSO

An adaptive variant of CPSOR is identified in [18] as the adaptive multi-swarm optimizer (AMSO). The goal of this variant is to make the number of populations adaptive. Although the number of sub-swarms in the CPSOR algorithm is variable, the author of AMSO identifies that the population is always restored to a constant number of particles which may not be appropriate for the environment. This new variant substitutes the population regeneration by estimating the number of particles required to accommodate the current landscape and to use that number in re-generation. The trigger monitors the rate at which the number of populations decrease during the search caused by the exclusion and convergence operators; reffered to as the drop rate. The trigger activates when the drop rate approaches zero and a minimum length of time δ has passed, over which the drop rate is calculated. In this framework, time is measured in function evaluations.

When triggered, an estimation for the ideal number of sub-populations is calculated based on the difference between the current number of surviving particles and the number at the last triggered interval. This allows the algorithm to initially inject

new particles even if all of the optima are found, but to note the second time around that the number of populations formed did not increase as a result and, therefore, not to inject any more particles the second time [18]. The intention of CPSOR and AMSO is to deal with environments where changes are difficult or impossible to detect therefore it does not use change detection. The goal of this specific adaptive behaviour is to deal with environments where the number of optima in the landscape changes over time and to increase overall efficiency in terms of function evaluations.

It is worth noting that CPSOR and AMSO have an altered personal best update operator, which is used to increase the speed of convergence [19, 18]. The operator works by taking particles who have bettered their $pBest$ and iteratively swaps each dimension of the new position with that of its swarm's best position, updating the swarm's best position if an improvement is yielded. AMSO alters this by probabilistically selecting a subset of the dimensions to check based on the distance between the swarm best and the new position in that given dimension, instead checking all dimensions.

5.3 Discussion

This section discusses the necessary considerations for the application of the algorithms reviewed in the previous section.

The problem with some of the algorithms presented above in terms of their multi-population techniques is that they rely on predefined constants on which the formation/behaviour of niches is based upon, i.e., the cloud radius of QSO and the species radius of SPSO [15]. This is an issue because it assumes that these characteristics are known about a given search space and that these characteristics are uniform throughout. This issue is also present in algorithms where recommended values for parameters are given but defined in terms of information such as number of peaks i.e CPSOR's $gsize$ and diversity threshold α [19].

Certain components were also designed for specific dynamic characteristics. The quantum and charged swarms are designed for environments with minor to modest movement from the peaks themselves because the tracking capability of the converging swarm is reliant on its non-converging radius [2]. These components may suffer in environments with higher spatial-change severities. Furthermore, it is seldom known

whether a given problem will have those characteristics or that they will be uniform throughout optimization.

It is important to recognize the components that the above algorithms use in order to make an effective selection for a given problem. Table 5.1 describes the PSO algorithms used by the individual swarms, as well as the diversity and memory mechanisms used to operate in dynamic environments. Table 5.2 categorizes the parameters of each algorithm by the behaviours which they influence.

Table 5.1: Algorithms' Sub-Swarm PSO and Dynamic Components

Algorithm	Sub-swarm PSO	Diversity Maintenance	Memory Update
CPSOR	inertia/$gbest_{learn}$	convergence, exclusion, overcrowding	re-evaluation
AMSO	inertia/$gbest_{problearn}$	convergence, exclusion, overcrowding	re-evaluation
SAMO	Quantum/Constriction	convergence, exclusion	sentry change detection
mQSO	Quantum/Constriction	convergence, exclusion	sentry change detection
mCSO	Charged/Constriction	convergence, exclusion	sentry change detection
SPSO	Constriction	implicit exclusion, overcrowding	re-evaluation
ANPSO	Constriction	implicit exclusion, overcrowding	re-evaluation

Table 5.2: Algorithm Categorized Parameters

Algorithm	Niching	Exclusion	Anti-Convergence	Population Size	Population Adaptation
CPSOR	$subSize$	$overlap$	$convergeThrsh$ (ϵ)	$gSize$	$proportion$ α
AMSO	$subSize$	$overlap$	$convergeThrsh$ (ϵ)	$gSize$ $min/maxPop$	$traceGap$ $increaseMultiplier$ $decreaseThrsh$
SAMO	-(fixed)	r_{excl}	r_{conv}	$subSize$ n_{excess}	n_{excess}
mCSO/mQSO	-(fixed)	r_{excl}	r_{conv}	$subSize$	-(fixed)
SPSO	$radius$ (r) $pmax$	$*inherent$	$pmax$	$popSize$	- (fixed)
ANPSO	$iter_{min}$ $iter_{length}$ $pmax$	$*inherent$	$pmax$	$popSize$	- (fixed)

This chapter summarized various niching PSO algorithms and classified their operators and parameters. The following chapter presents a statistical comparison of these algorithms in terms of their niching performance.

Chapter 6

Algorithm Performance Comparison

This chapter provides an experimental analysis of the different dynamic niching techniques reviewed in the previous chapter. A brief discussion of past comparative work, as well as some improvements introduced in this work, form the basis for this study. A statistical comparison is performed to rank the algorithms on various configurations of the benchmark functions reviewed in Chapter 2. The niching performance of each algorithm is also analysed based on scalability as well as the landscape features characterized in Chapter 3.

6.1 Previous Comparative Studies

For the algorithms reviewed Chapter 5, most works which introduce an algorithm also provide a comparative analysis against existing peer algorithms [2, 4, 15, 17, 18, 19]. The analysis performed in many of these articles share a number of limitations in terms of the experimentation. One such limitation is the attempt to compare the algorithms as a whole, rather than the components of which they are composed of. As shown in Table 5.1 of the previous chapter, multi-modal PSO algorithms are composed of smaller components or mechanisms which contribute to its overall behaviour. The problem with comparing the algorithms based on their overall behaviours is the lack of information the comparison provides in terms of the contribution to performance of the individual components.

Measuring niching performance is another area of limited experimentation. When using benchmark functions such as the MPB [6] and GDBG [24] with known optima,

a commonly used metric is the standard error measurement. Error is the difference between the optimal solution and the best found solution in terms of a given fitness function. Error only measures an algorithm's ability to locate a single, global optimum, yet it is often the sole criteria used to compare niching algorithms. Niching performance should include the number of distinct optima found as well as the accuracy achieved for each of those optima; these two concepts are referred to in this chapter as coverage and accuracy, respectively. In [18] a tracking ratio ($tRatio$) is defined as the number of peaks tracked over the total number of peaks, where a tracked peak is one that has particles within a Euclidean threshold of its center. Although this measure does consider more than just a single optimum, it does not measure accuracy. This chapter uses a modified version of the $tRatio$ and introduces a similar metric $PeakDist$ to include accuracy.

A third oversight in comparing algorithms is optimizing their parameters. It is often the case where the parameters of an algorithm being introduced in a given study are empirically established for the functions being tested. The process of establishing appropriate values for the parameters can also be referred to as tuning. In most experimental comparisons, the peer algorithms are simply assigned parameter configurations established by their respective studies. This scenario can be considered valid under the assumption that the test function implementations are identical and that the assigned configurations correspond to similar tests to the ones being carried out; however these assumptions are often not considered where configurations presented in original work is interpreted as being general. An example of this type of oversight can be found in [22] where an empirical study is conducted under a variety environments using the MPB benchmark function. The algorithm configurations used were derived from their originating work although most of those studies [2, 16, 17, 4] did not consider some of the environments being tested, e.g., environments with a dynamic number of optima. When applying existing algorithms to new environments, it is necessary to tune the algorithms before hand, or be self-adaptive during a run, in order to provide a fair comparison.

6.2 Experimental Setup

The experiments conducted in this chapter are to provide a comparative analysis of the seven reviewed niching PSO algorithms in the context of the six benchmark functions reviewed in Chapters 2 and 3. Specifically, the analysis will include a

statistical comparison of the algorithms and an examination of the potential effects that different landscape characteristics have on niching. The algorithms were tested in 5 dimensions under various peak-counts and shift severities. In an attempt to quantify the average displacement by a peak under both the shifting (MPB) [6] and rotation (GDBG) [24] frameworks, preliminary tests were conducted on the two operators. The tests revealed that the rotation operator yields a high variance in peak shift distances under both the small and large change seen in Equations 2.11 and 2.12, whereas the shifting operator from the MPB provides exactly the defined shift regularly. For this reason, the rotation operator was replaced by the shift operator for the GDBG composition functions. The shift values in this study were interpreted as a proportion of the range of the function's bounds due to the different bounds used by the MPB and GDBG.

In order to isolate the niching components of the algorithms, the PSO model used by the sub-swarms of the different algorithms were replaced by the standard inertia-weight model reviewed in Chapter 4. The inertia weight w used was 0.729 and the social and cognitive coefficients, c_1 and c_2, were both set to 1.496 taken from studies in [28]. These values aren't very important to this study, just that they are the same for all algorithms. Change detection was also replaced for the re-evaluation model in the multi-swarm algorithms (i.e., SAMO, mQSO, and mCPSO) as it is used in the other four algorithms.

6.2.1 Full-Factorial Design

The full-factorial design (FFD) [57, 56] is employed in this study to address the issue of tuning the algorithms. The FFD method is comprised of selecting an appropriate range of values for each parameter and testing every combination of those values. Parameters dealing with specific Euclidean distances such as the species radius r of SPSO [15] and the quantum cloud radius r_{cloud} of mQSO [2] were interpreted as proportions of the maximum Euclidean distance possible (ed_{max}, Equation 6.1) of the function's domain. Again, this was done to facilitate testing across the different bounds used by the MPB and GDBG.

$$ed_{max} = \sqrt{range^2 * d} \tag{6.1}$$

When selecting the parameter ranges, considerations were made to ensure fairness in terms of number of particles used, as well as number of configurations tested: for

each of the environments, 160 configurations allowing between 200 and 800 particles were tested per algorithm.

6.2.2 Statistical Comparisons

A statistical comparison model introduced in [60] is adopted to compare the niching algorithms. The model uses a non-parametric test such as the Mann-Whitney-U, in order to perform a pairwise comparison of algorithm performances on a single metric. An algorithm is awarded a win for each of the pairwise comparisons in which it is statistically better; similarly, the losing algorithm is awarded a loss. If there is no significant difference, no wins/losses are assigned. Finally, observations can be made based on the comparison of total wins vs. losses between algorithms.

This model is used in two phases of this study. The first use is to compare the different configurations of each separate algorithm in order to chose the best one for a given environment. The second use is to compare the selected algorithm configurations for each environment, thus comparing the algorithms. The Mann-Whitney-U is used as the statistical test with a significance level of 0.05.

6.2.3 Performance Metrics

The $offline_{error}$ is the average error sampled at a given interval and is used to measure an algorithm's ability to find a global optimum (accuracy) as well as how fast it does so (efficiency). Offline error is calculated as follows:

$$offline_{error} = \frac{\sum_{i\%interval_m=0}^{evals} (optimal(i) - best(i))}{evals/interval_m} \tag{6.2}$$

where $evals$ is the total number of evaluations allowed and $interval_m$ is the measurement interval in function evaluations. For example, two algorithms can find the same optimum before the change interval, but the algorithm that finds the particular optimum first will have a lower $offline_{error}$ since more of it's samples will have a lower error in the calculating the average. The interval is often measured in algorithm iterations. However a fairer base, especially in dynamic environments, would be function evaluations due to the ambiguity of an iteration, i.e., the number of function evaluations per iteration varies depending between algorithms.

The *peakCover* measure is an adaptation of the *tRatio* mentioned above. Instead of the threshold t being a hard Euclidean distance, it is instead a proportion of ed_{max}.

$$peakCover = P_c/P \tag{6.3}$$

$$P_c = \sum_{dist(i)<t}^{P} 1 \tag{6.4}$$

Equation 6.3 describes $PeakCover$ as the number of peaks covered P_c over total peaks P. Equation 6.4 defines the peak cover calculation. $PeakCover$ is used to determine the proportion of optima that are being tracked, where $dist(i)$ is the Euclidean distance between optima i and the closes particle and t represents the cut-off threshold of $dist(i)$ for a optimum i to be considered tracked. $PeakCover$ does not consider the accuracy of that tracking and should be used in conjunction with the $PeakDist$ measure.

The *peakDist*, expressed in Equation 6.5, measures the Euclidean distance between a peak and the closest particle if the distance is smaller than a given threshold. The distance is expressed as a proportion of the threshold which is the same way as in *peakCover*.

$$PeakDist = \sum_{dist(i)<t}^{P} (dist(i))/P_c \tag{6.5}$$

$PeakDist$ is expressed as the average distance of all covered peaks and is used to observe the accuracy of the multi-peak tracking but does not reveal the number of peaks covered. Note that the $offline$ modifier is applied to the $PeakCover$ and $PeakDist$ to compare performance between runs.

6.3 Results and Analysis

The analysis in this section is presented in two sections. The first is a detailed statistical comparison of the algorithms' performances based on the $Error$, $PeakCover$, and $PeakDist$ metrics. The second section reviews scalability of each niching algorithm with respect to modality and shift severity for the $PeakCover$, and $PeakDist$ metrics. For each of the metrics in both sections, a set of results is presented using the algorithm configurations that performed best with respect to that particular metric.

For each of these sets, the results for the other metrics are also shown to evaluate performance trade-offs between the objectives of the metrics.

6.3.1 Statistical Comparison of Algorithms

Table 6.1: Overall wins and losses for best algorithm configurations selected based on $offline_{Error}$

Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Error	Wins	307	193	243	89	0	61	142
	Losses	3	101	60	162	313	243	153
	Diff	304	92	183	-73	-313	-182	-11
	Rank	**1**	3	2	5	7	6	4
PeakCover	Wins	204	297	228	68	173	4	113
	Losses	103	12	77	251	132	318	194
	Diff	101	285	151	-183	41	-314	-81
	Rank	3	**1**	2	6	4	7	5
PeakDist	Wins	305	121	193	20	227	167	28
	Losses	4	191	103	293	61	124	285
	Diff	301	-70	90	-273	166	43	-257
	Rank	**1**	5	3	7	2	4	6

Table 6.1 summarizes comparison scores between the algorithms' best performing configurations in terms of $offline_{Error}$ for all of the experiments, overall functions and every modality/shift severity. The AMSO algorithm achieved the best accuracy both in terms of peaks tracked as well as the global best peak. ANPSO performs best in terms of number of optima tracked, doing so at the expense of accuracy as it ranks 5^{th} in terms of $PeakDist$. Similarly, CPSOR also ranks higher in $PeakCover$ and lower in terms of accuracy. The mQSO algorithm ranks second in $PeakDist$ but last in terms of error, suggesting that the global optima are not being tracked. Overall, the multi-swarm algorithms (mQSO, mCPSO, SAMO) rank the worst in terms of the error metric, which could be caused by the removal of the change detection mechanism. This is also supported by the low ranks achieved in the number of optima tracked.

Table 6.2 shows a comparison similar to the one in Table 6.1, however the algorithm configurations chosen were those optimized for maximal peaks tracked ($offline_{PeakCover}$). CPSOR achieved the best rank in this category, while ranking 2^{nd} in error and 4^{th}

Table 6.2: Overall wins and losses for best algorithm configurations selected based on $offline_{PeakCover}$

Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Error	Wins	312	208	250	68	36	17	175
	Losses	5	94	61	223	261	284	138
	Diff	307	114	189	-155	-225	-267	37
	Rank	**1**	3	2	5	6	7	4
PeakCover	Wins	236	249	292	57	128	0	146
	Losses	76	63	24	265	188	324	168
	Diff	160	186	268	-208	-60	-324	-22
	Rank	3	2	**1**	6	5	7	4
PeakDist	Wins	245	70	125	118	251	257	4
	Losses	60	239	184	197	50	23	317
	Diff	185	-169	-59	-79	201	234	-313
	Rank	3	6	4	5	2	**1**	7

in peak accuracy. AMSO shows a similar results to the previous comparison, however its rank for overall peak accuracy dropped to 3^{rd}. SPSO and ANPSO's rankings also remained very similar, suggesting that their parameters have little effect on the trade-off between accuracy and coverage. Once again, mQSO and SAMO show higher rankings in *PeakDist*. However, looking at the other two measures, the accuracy achieved seems to be on a low number of peaks tracked which do not include the global optimum since the proportion of peaks tracked is lower.

Table 6.3 shows yet another comparison, where the algorithm configurations chosen were those optimized for the average accuracy of all peaks tracked ($offline_{PeakDist}$). Similar to the results in Table 6.1, AMSO achieved rank 1 in both *PeakDist* and error, but its *PeakCover* rank drops to second-last position. This suggests that the algorithm's performance can be tuned to trade coverage for accuracy and vice-versa. CPSOR also shows a similar trend, having it's lowest *PeakCover* ranking of all three tables. The results suggest that the adaptive re-diversification operator of AMSO is better suited for accuracy, while the original operator it replaced in CPSOR is better suited for peak coverage. SPSO ranks last in overall peak accuracy with no wins. ANPSO and SPSO rankings are relatively unchanged between the three tables, reinforcing the assumption that their performance has limited sensitivity to their parameters. mQSO has the best overall performance out of the multi-swarm algorithms, ranking 4^{th} in accuracy and 2^{nd} in coverage. In all three tables, SAMO

Table 6.3: Overall wins and losses for best algorithm configurations selected based on $offline_{PeakDist}$

Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Error	Wins	297	219	249	21	14	65	168
	Losses	11	79	49	250	273	225	146
	Diff	286	140	200	-229	-259	-160	22
	Rank	**1**	3	2	6	7	5	4
PeakCover	Wins	95	312	165	99	225	2	171
	Losses	209	7	138	201	76	317	121
	Diff	-114	305	27	-102	149	-315	50
	Rank	6	**1**	4	5	2	7	3
PeakDist	Wins	296	91	198	66	180	198	0
	Losses	10	212	86	247	97	57	320
	Diff	286	-121	112	-181	83	141	-320
	Rank	**1**	5	3	6	4	2	7

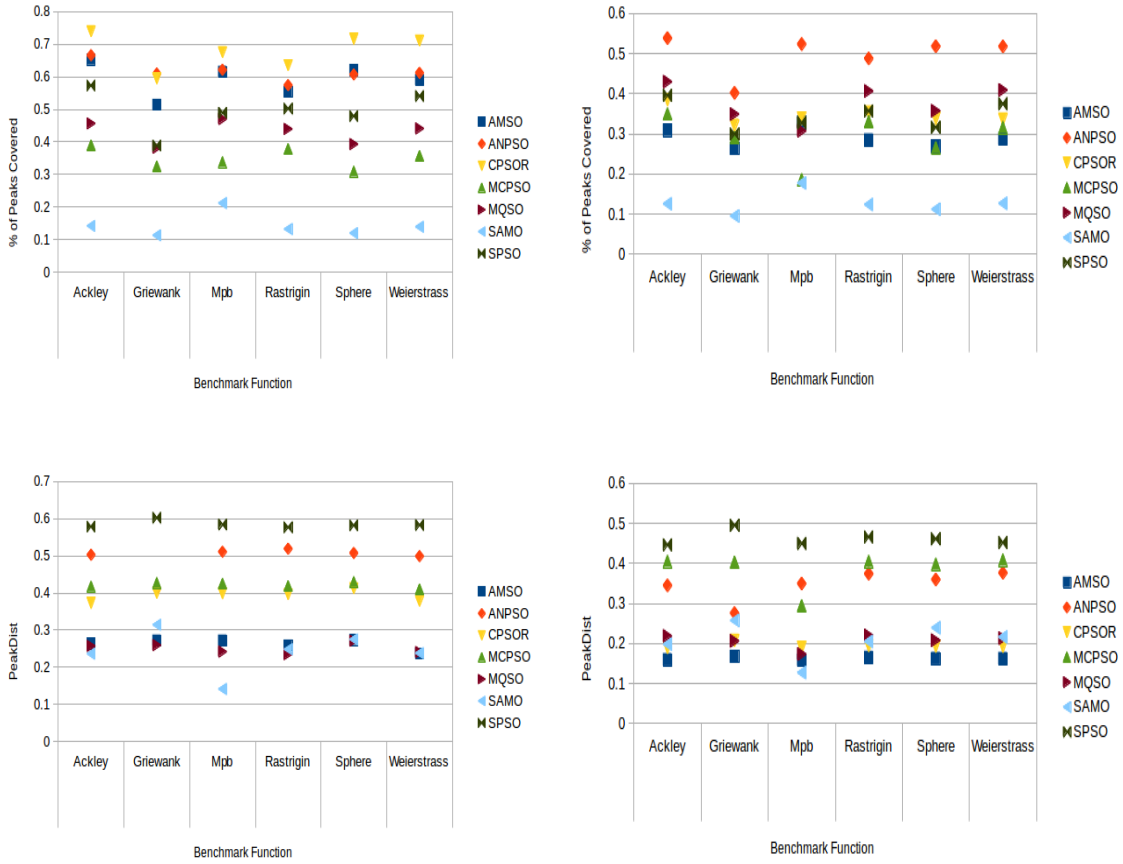
scores last in coverage which dismisses its higher rankings in the *PeakDist* measure since it is tracking the fewest peaks.

Appendix B contains further classification of the data in Tables 6.1, 6.2, and 6.3 based on optimization function, modality, and shift severities. In general, the data follows the same trends described in this chapter with the exception of Tables B.7 and B.8; both show the AMSO algorithm achieving rank 1 across all three measures in higher shift severities. This suggests that the AMSO algorithm has better scalability in abrupt and chaotic environments.

6.3.2 Niching and Landscape Features

Figure 6.1 shows the average performance in terms of *PeakCover* and *PeakDist* for each of the benchmark functions used, summarized from Tables 6.6 and 6.7. The plots on the left represent the algorithm configurations that performed best for peak coverage, while the two on the right represent the best configurations in terms of peak accuracy. These plots confirm that some of the algorithms can be tuned to trade off between accuracy and coverage better than others. Table 6.4 calculates the average coverage-accuracy trade-off ratio of each algorithm by comparing the differences of the metric readings from the two configurations. AMSO has the largest ratio of $2.98 : 1$, which means its coverage performance can be traded for performance in accuracy at

Figure 6.1: Average $offline_{PeakCover}$ and $offline_{PeakDist}$ for the algorithms (left plots: optimized for $PeakCover$, right plots: optimized for $PeakDist$).



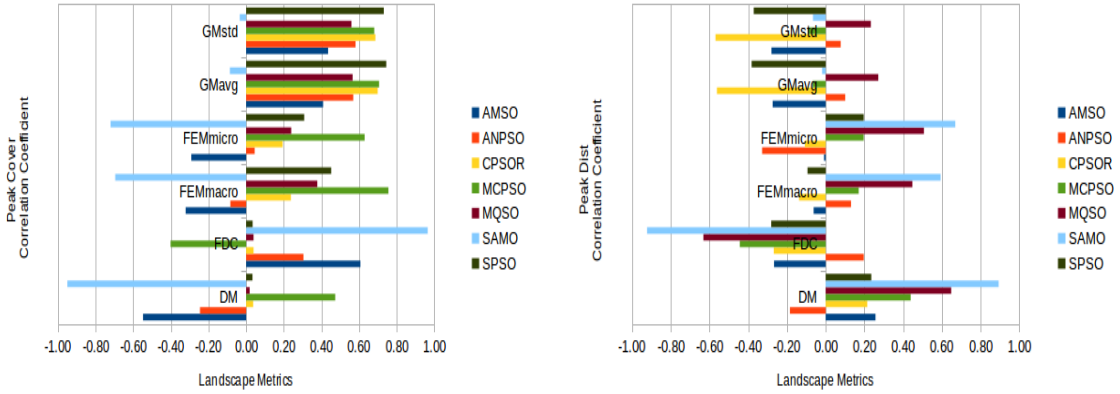
a rough rate of 3 to 1. CPSOR and MCPSO have similar ratios although CPSOR has a much larger tuneable range, shown by the calculated differences of the two metrics. The multi-swarm algorithms have low ranges, meaning their parameters are not tuneable for different coverage-accuracy performance profiles. ANPSO and SPSO have a near 1 to 1 ratios and ranges of roughly $10\% - 15\%$ which can be interpreted as moderate tuneability.

The charts in Figure 6.1 do not show many general links between performance and benchmark functions. In the overall performance averages shown, every algorithm remained within a rough $10\% - 15\%$ range across all the benchmark functions. To test for individual effects of the landscape characteristics reviewed in Chapter 2 on the niching algorithms, their performance is compared to the landscape characterization results obtained in Chapter 3. Using the landscape metric results from each

Table 6.4: Average coverage-accuracy trade-off ratios

Algortithm	Diff-Coverage	Diff-Accuracy	Ratio
AMSO	0.3008	0.1008	2.98:1
ANPSO	0.1166	0.1448	0.8:1
CPSOR	0.3334	0.1988	1.68:1
MCPSO	0.0597	0.0353	1.69:1
MQSO	0.0537	0.0444	1.21:1
SAMO	0.0168	0.0351	0.47:1
SPSO	0.1504	0.1222	1.23:1

Figure 6.2: Correlation coefficients calculated between the niching performance for each algorithm ($offline_{PeakCover}$, $offline_{PeakDist}$) and the landscape metric readings from each of the benchmark functions measured in Chapter 3.



benchmark function, a correlation-coefficient is calculated to evaluate the connection between a given landscape feature and the niching performance metrics. The 5-dimensional characterization data was used to match experimentation done on the niching algorithms.

The coefficients listed in Table 6.5 were calculated using performance and characterization results matched by the function on which they were obtained. The niching performance data used was an average of the two sets; one set optimized for coverage, the other for accuracy. Figure 6.2 displays the peak coverage and accuracy correlations for every algorithm/landscape metric pair. The peak coverage (left plot) shows a significant positive correlation with the gradient measures (GM) for all algorithms except SAMO. This correlation suggests that the niching techniques can locate more peaks when the gradients are steeper.

Table 6.5: Correlation coefficients calculated between the niching performance for each algorithm ($offline_{PeakCover}$, $offline_{PeakDist}$) and the Landscape Metric readings from each of the benchmark functions.

Performance Metric	Landscape Metric	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
PeakCover	DM	-0.5477	-0.2452	0.0373	0.4734	0.0190	-0.9498	0.0337
	FDC	0.6072	0.3047	0.0395	-0.4013	0.0394	0.9648	0.0346
	FEMmacro	-0.3214	-0.0838	0.2381	0.7560	0.3784	-0.6954	0.4521
	FEMmicro	-0.2914	0.0454	0.1941	0.6297	0.2397	-0.7199	0.3086
	GMavg	0.4090	0.5693	0.6989	0.7065	0.5658	-0.0864	0.7447
	GMstd	0.4356	0.5813	0.6867	0.6809	0.5596	-0.0339	0.7316
PeakDist	DM	0.2575	-0.1844	0.2150	0.4390	0.6483	0.8933	0.2356
	FDC	-0.2664	0.1966	-0.2680	-0.4435	-0.6312	-0.9218	-0.2811
	FEMmacro	-0.0620	0.1312	-0.1377	0.1708	0.4478	0.5935	-0.0937
	FEMmicro	-0.0086	-0.3280	-0.1076	0.1963	0.5076	0.6695	0.1965
	GMavg	-0.2739	0.1012	-0.5613	-0.0672	0.2720	-0.0184	-0.3824
	GMstd	-0.2799	0.0775	-0.5687	-0.0919	0.2334	-0.0655	-0.3714

The FDC metric, which measures general searchability, shows a significant negative correlation with accuracy (right plot) for most algorithms as well. Since the $PeakDist$ measurement is a minimising one, the negative correlation suggests that when the FDC rises, the accuracy improves. This is to be expected; as searchability improves, it should enable individual swarms to better approach the local optima. A similar, positive correlation with the dispersion metric (DM) is also observed in the accuracy plot. This is also expected as multi-funnel landscapes, denoted by higher DM values, should cause higher $PeakDist$ readings, translating into a decrease in accuracy. The ruggedness metric (FEM) shows a moderate positive correlation for the multi-swarm algorithms in terms of their accuracy performance, suggesting that it has a significant, negative effect.

6.3.3 Modality and Shift Scalability

In this section the niching performances of the algorithms are analysed and compared based on their scalability in modality and shift severity. The data used is the average performance for each benchmark function under the varying peak counts and movements, summarized in Tables 6.6 and 6.7. For the modality experiments, each environment uses a shift value of of 0.01. Similarly, under the shift experiments, each environment uses a peak count of 10.

Figure 6.3 illustrates the coverage and accuracy plots for the niching algorithms in terms of environment modality; optimized for peak coverage on the left and accuracy on the right. Comparing the two configurations in terms of coverage overall,

Figure 6.3: Average $offline_{PeakCover}$ and $offline_{PeakDist}$ for various modalities (left plots: optimized for $PeakCover$, right plots: optimized for $PeakDist$).

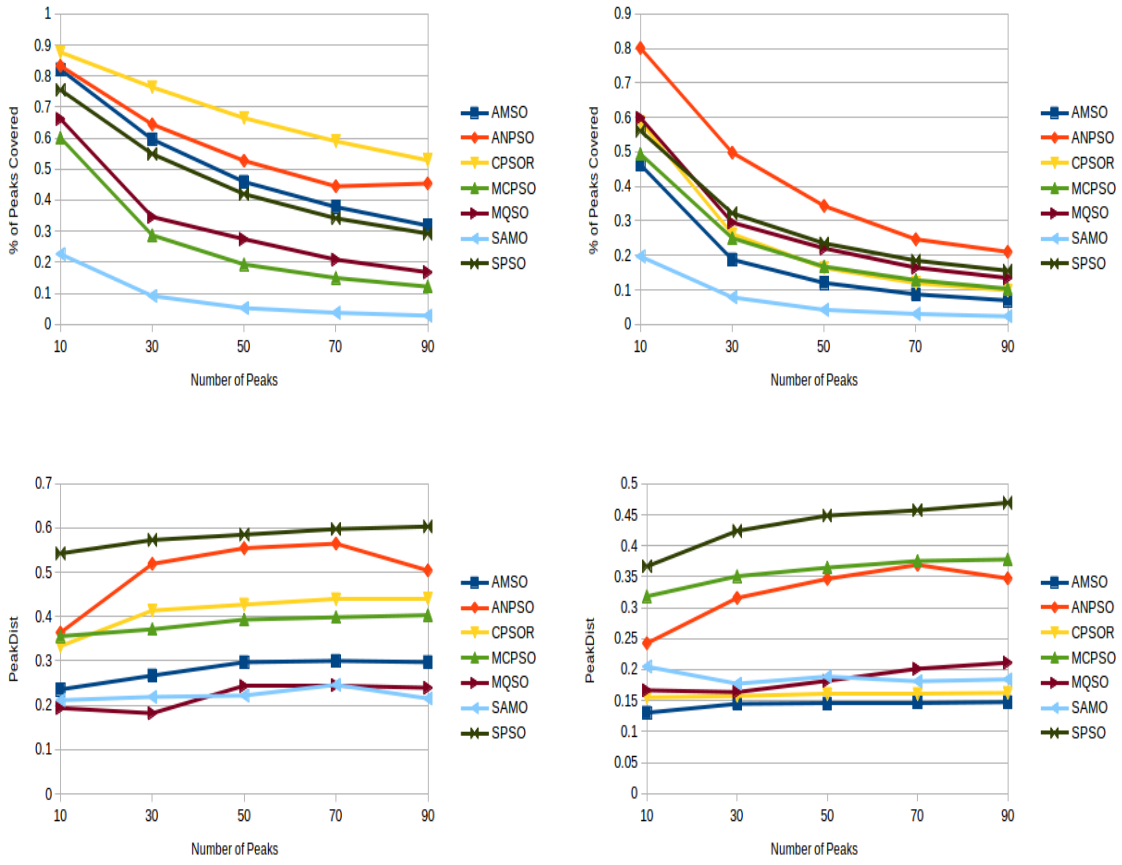


Table 6.6: Average $offline_{PeakCover}$ and $offline_{PeakDist}$ for each of the benchmark functions. Algorithm configurations optimized for $offline_{PeakDist}$

Problem	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Ackley	PeakCover	Avg	0.3085	0.5389	0.3848	0.3490	0.4303	0.1254	0.3961
		StdDev	0.1089	0.1929	0.1450	0.2203	0.2492	0.0835	0.1767
	PeakDist	Avg	0.1587	0.3452	0.1899	0.4050	0.2187	0.1981	0.4465
		StdDev	0.2084	0.1908	0.2421	0.2460	0.2850	0.2901	0.1910
Griewank	PeakCover	Avg	0.2633	0.4022	0.3210	0.2905	0.3495	0.0948	0.2995
		StdDev	0.1005	0.1655	0.1379	0.1785	0.2024	0.0704	0.1478
	PeakDist	Avg	0.1671	0.2758	0.2073	0.4030	0.2064	0.2575	0.4955
		StdDev	0.2194	0.2136	0.2585	0.2438	0.2731	0.3506	0.1985
Mpb	PeakCover	Avg	0.3283	0.5241	0.3381	0.1855	0.3076	0.1772	0.3277
		StdDev	0.1168	0.1884	0.1360	0.1442	0.1910	0.0903	0.1754
	PeakDist	Avg	0.1573	0.3499	0.1887	0.2942	0.1736	0.1268	0.4502
		StdDev	0.2284	0.1925	0.2355	0.3480	0.2462	0.1991	0.2100
Rastrigin	PeakCover	Avg	0.2844	0.4881	0.3539	0.3305	0.4070	0.1234	0.3571
		StdDev	0.1003	0.1970	0.1375	0.2171	0.2430	0.0783	0.1662
	PeakDist	Avg	0.1646	0.3744	0.1950	0.4048	0.2203	0.2052	0.4663
		StdDev	0.2126	0.1931	0.2420	0.2502	0.2885	0.2951	0.1899
Sphere	PeakCover	Avg	0.2686	0.5186	0.3379	0.2651	0.3576	0.1117	0.3161
		StdDev	0.1074	0.1878	0.1304	0.1675	0.2004	0.0789	0.1615
	PeakDist	Avg	0.1621	0.3600	0.1931	0.3979	0.2081	0.2392	0.4616
		StdDev	0.2212	0.1904	0.2426	0.2730	0.2621	0.3177	0.2056
Weierstrass	PeakCover	Avg	0.2869	0.5182	0.3364	0.3154	0.4096	0.1264	0.3750
		StdDev	0.1021	0.1923	0.1369	0.2077	0.2389	0.0874	0.1653
	PeakDist	Avg	0.1619	0.3767	0.1929	0.4084	0.2132	0.2160	0.4523
		StdDev	0.2051	0.1861	0.2351	0.2480	0.2812	0.2995	0.1885

the ability for most algorithms to scale with modality is greatly compromised when optimized for accuracy. SPSO, AMSO, and CPSOR's coverage drops below 30% of peaks by the time peak count reaches 30-peaks. This drop continues to the 90-peak mark where most of the algorithms are at 15% peak cover or below. This translates roughly to no more than 10-15 peaks being tracked by these three algorithms in any given modality. This contrasts significantly to the configurations on the left, where CPSOR is still over 50% coverage at the 90-peak environment with SPSO and AMSO around 30% coverage. ANPSO's drop rate is slower than the last three and is still able to track a majority of the peaks in the 10 to 30-peak environments. As previously mentioned, the multi-swarm algorithms (mQSO, mCPSO, and SAMO) remained relatively unchanged between both configurations.

Looking at the accuracy charts, overall accuracy remained relatively stable with the increase in modality, compared to peak coverage. SPSO and ANPSO demonstrate

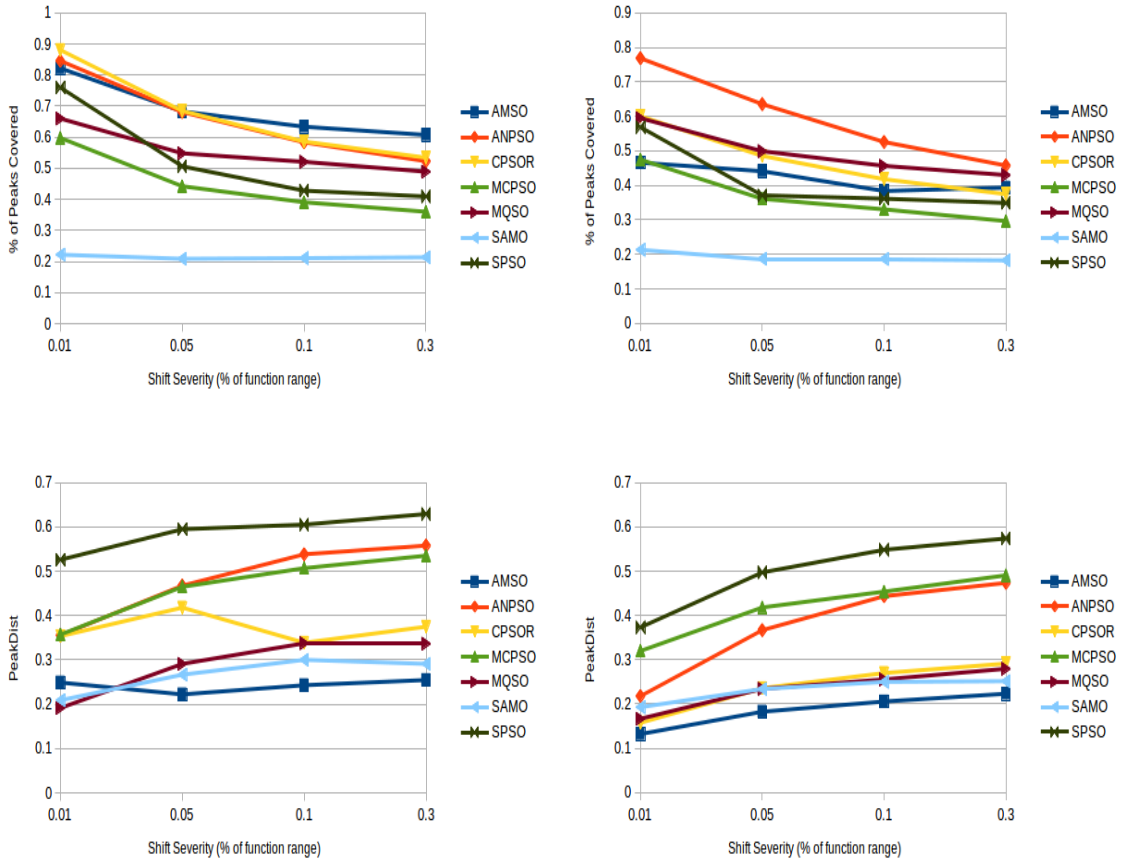
Table 6.7: Average $offline_{PeakCover}$ and $offline_{PeakDist}$ for each of the benchmark functions. Algorithm configurations optimized for $offline_{PeakCover}$

Problem	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Ackley	PeakCover	Avg	0.6520	0.6659	0.7396	0.3889	0.4569	0.1421	0.5731
		StdDev	0.2480	0.2721	0.2686	0.2112	0.2638	0.1075	0.2763
	PeakDist	Avg	0.2643	0.5027	0.3730	0.4170	0.2574	0.2362	0.5785
		StdDev	0.2442	0.1535	0.2100	0.2179	0.2789	0.3188	0.1527
Griewank	PeakCover	Avg	0.5139	0.6097	0.5949	0.3243	0.3822	0.1133	0.3889
		StdDev	0.2152	0.2404	0.2474	0.1753	0.2132	0.0924	0.2024
	PeakDist	Avg	0.2719	0.4123	0.4001	0.4267	0.2597	0.3142	0.6023
		StdDev	0.2545	0.1895	0.2195	0.2203	0.2435	0.3581	0.1572
Mpb	PeakCover	Avg	0.6152	0.6212	0.6756	0.3378	0.4702	0.2121	0.4893
		StdDev	0.2400	0.2595	0.2710	0.1731	0.2306	0.1152	0.2388
	PeakDist	Avg	0.2719	0.5108	0.3989	0.4244	0.2420	0.1412	0.5838
		StdDev	0.2441	0.1599	0.2259	0.2211	0.2089	0.1993	0.1531
Rastrigin	PeakCover	Avg	0.5537	0.5741	0.6352	0.3779	0.4399	0.1324	0.5021
		StdDev	0.2232	0.2602	0.2582	0.2039	0.2448	0.1006	0.2568
	PeakDist	Avg	0.2587	0.5189	0.3963	0.4182	0.2352	0.2483	0.5766
		StdDev	0.2517	0.1581	0.2130	0.2182	0.2640	0.3223	0.1632
Sphere	PeakCover	Avg	0.6206	0.6070	0.7160	0.3088	0.3934	0.1201	0.4793
		StdDev	0.2329	0.2482	0.2680	0.1549	0.2423	0.0997	0.2335
	PeakDist	Avg	0.2735	0.5074	0.4130	0.4293	0.2715	0.2751	0.5818
		StdDev	0.2422	0.1554	0.2057	0.2113	0.2656	0.3564	0.1560
Weierstrass	PeakCover	Avg	0.5897	0.6118	0.7111	0.3566	0.4412	0.1395	0.5407
		StdDev	0.2111	0.2645	0.2679	0.1958	0.2495	0.1076	0.2657
	PeakDist	Avg	0.2364	0.4989	0.3787	0.4094	0.2410	0.2380	0.5827
		StdDev	0.2266	0.1658	0.2109	0.2233	0.2703	0.3185	0.1535

faster drop rates in accuracy under both configurations, which is expected since they have slower drop rates in coverage; the more peaks covered, the less accurately they are covered. When tuned for accuracy, CPSOR and AMSO maintain a constant level of around 0.17 which is the best scores obtained.

Figure 6.4 illustrates the coverage and accuracy plots for the niching algorithms in terms of peak shift severity; optimized for peak coverage on the left and accuracy on the right. The effects of peak displacement are not as uniform across the set of niching as seen in the modality experiments. When optimized for coverage, CPSOR and ANPSO achieve higher peak coverage in lower shift severities. However, AMSO maintains a slower drop rate, making it a better performer for higher peak displacements. AMSO also maintains the best accuracy under both configuration sets. As seen with modality, most algorithms achieve similar coverage performance when optimized for accuracy with ANPSO performing significantly better.

Figure 6.4: Average $offline_{PeakCover}$ and $offline_{PeakDist}$ for various shift severities (left plots: optimized for $PeakCover$, right plots: optimized for $PeakDist$).



In conclusion, a general correlation was found between coverage performance and the gradient measures, suggesting that higher gradients improved ability to locate more optima. Another correlation was found, albeit weaker, between the searchability metric and accuracy performance; suggesting that lower searchability readings inhibit an algorithms ability to approach local optima. When comparing the algorithms, none of the algorithms in this studied dominated all others in both coverage and accuracy using a single configuration. A clear trade-off between accuracy and coverage was demonstrated by every algorithm and measured in Table 6.5. Overall, the shift severities had more effect on accuracy performance while modality had greater influence over peak coverage.

Table 6.8: Average $offline_{PeakCover}$ and $offline_{PeakDist}$ for each modality/shift severity. Algorithm configurations optimized for $offline_{PeakDist}$

		Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Peaks	10	PeakCover	Avg	0.4627	0.8009	0.5874	0.4939	0.5987	0.1968	0.5612
			StdDev	0.1452	0.2216	0.1991	0.2611	0.2734	0.1256	0.2282
		PeakDist	Avg	0.1304	0.2425	0.1551	0.3183	0.1666	0.2048	0.3666
			StdDev	0.1900	0.2014	0.2392	0.2607	0.2254	0.2967	0.2273
	30	PeakCover	Avg	0.1874	0.4978	0.2619	0.2498	0.2950	0.0781	0.3216
			StdDev	0.0624	0.1597	0.0906	0.1407	0.1991	0.0404	0.1409
		PeakDist	Avg	0.1447	0.3157	0.1573	0.3508	0.1636	0.1772	0.4239
			StdDev	0.1972	0.1670	0.2217	0.2358	0.2459	0.2563	0.1808
	50	PeakCover	Avg	0.1200	0.3431	0.1635	0.1673	0.2201	0.0419	0.2347
			StdDev	0.0396	0.1171	0.0564	0.0965	0.1513	0.0226	0.1040
		PeakDist	Avg	0.1450	0.3465	0.1605	0.3646	0.1818	0.1887	0.4485
			StdDev	0.1931	0.1534	0.2133	0.2278	0.2439	0.2711	0.1581
	70	PeakCover	Avg	0.0869	0.2464	0.1193	0.1279	0.1643	0.0299	0.1847
			StdDev	0.0276	0.0869	0.0410	0.0748	0.1225	0.0156	0.0835
		PeakDist	Avg	0.1466	0.3690	0.1605	0.3753	0.2013	0.1811	0.4572
			StdDev	0.1893	0.1402	0.2108	0.2235	0.2789	0.2675	0.1492
	90	PeakCover	Avg	0.0692	0.2098	0.0982	0.1031	0.1343	0.0231	0.1552
			StdDev	0.0219	0.0751	0.0324	0.0609	0.1007	0.0132	0.0705
		PeakDist	Avg	0.1475	0.3471	0.1624	0.3778	0.2111	0.1842	0.4693
			StdDev	0.1856	0.1497	0.2152	0.2198	0.2907	0.2723	0.1382
Shift	0.01	PeakCover	Avg	0.4658	0.7684	0.6003	0.4745	0.5948	0.2129	0.5687
			StdDev	0.1418	0.2156	0.1888	0.2675	0.2739	0.1279	0.2343
		PeakDist	Avg	0.1320	0.2178	0.1574	0.3199	0.1665	0.1928	0.3732
			StdDev	0.1870	0.2011	0.2268	0.2725	0.2263	0.2866	0.2313
	0.05	PeakCover	Avg	0.4410	0.6354	0.4854	0.3608	0.4990	0.1870	0.3709
			StdDev	0.1748	0.2694	0.2020	0.2695	0.2882	0.1265	0.1958
		PeakDist	Avg	0.1825	0.3669	0.2358	0.4183	0.2343	0.2340	0.4974
			StdDev	0.2481	0.2426	0.2644	0.3187	0.2989	0.3243	0.2300
	0.1	PeakCover	Avg	0.3835	0.5258	0.4178	0.3303	0.4565	0.1863	0.3612
			StdDev	0.1638	0.2741	0.2168	0.2626	0.2842	0.1312	0.2125
		PeakDist	Avg	0.2056	0.4437	0.2699	0.4541	0.2556	0.2499	0.5485
			StdDev	0.2715	0.2452	0.2910	0.3178	0.3128	0.3261	0.2318
	0.3	PeakCover	Avg	0.3934	0.4574	0.3745	0.2962	0.4299	0.1824	0.3489
			StdDev	0.1770	0.2664	0.2085	0.2695	0.2942	0.1302	0.2197
		PeakDist	Avg	0.2231	0.4737	0.2913	0.4907	0.2796	0.2516	0.5742
			StdDev	0.2808	0.2490	0.3012	0.3370	0.3314	0.3273	0.2285

Table 6.9: Average $offline_{PeakCover}$ and $offline_{PeakDist}$ for each modality/shift severity. Algorithm configurations optimized for $offline_{PeakCover}$

		Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Peaks	10	PeakCover	Avg	0.8208	0.8325	0.8764	0.6008	0.6618	0.2262	0.7552
			StdDev	0.2563	0.2478	0.2361	0.2424	0.2661	0.1395	0.3061
		PeakDist	Avg	0.2356	0.3640	0.3335	0.3558	0.1935	0.2112	0.5427
			StdDev	0.2429	0.1849	0.2177	0.2151	0.2211	0.2854	0.1713
	30	PeakCover	Avg	0.5958	0.6441	0.7638	0.2868	0.3459	0.0913	0.5489
			StdDev	0.2330	0.2747	0.2812	0.1432	0.2184	0.0606	0.2425
		PeakDist	Avg	0.2669	0.5191	0.4139	0.3715	0.1817	0.2187	0.5731
			StdDev	0.2428	0.1374	0.1924	0.1981	0.2331	0.2917	0.1234
	50	PeakCover	Avg	0.4589	0.5270	0.6645	0.1928	0.2742	0.0522	0.4194
			StdDev	0.2083	0.2439	0.2742	0.1020	0.1832	0.0369	0.1915
		PeakDist	Avg	0.2969	0.5545	0.4272	0.3934	0.2433	0.2218	0.5851
			StdDev	0.2384	0.1113	0.1832	0.1872	0.2416	0.2978	0.1059
	70	PeakCover	Avg	0.3779	0.4445	0.5898	0.1493	0.2084	0.0371	0.3413
			StdDev	0.1741	0.2124	0.2592	0.0798	0.1398	0.0248	0.1587
		PeakDist	Avg	0.3000	0.5648	0.4391	0.3986	0.2453	0.2461	0.5977
			StdDev	0.2330	0.0996	0.1747	0.1781	0.2423	0.2971	0.0915
	90	PeakCover	Avg	0.3180	0.4537	0.5285	0.1211	0.1675	0.0280	0.2922
			StdDev	0.1482	0.1963	0.2383	0.0653	0.1103	0.0199	0.1362
		PeakDist	Avg	0.2972	0.5042	0.4404	0.4034	0.2393	0.2151	0.6034
			StdDev	0.2286	0.1099	0.1700	0.1750	0.2402	0.3003	0.0830
Shift	0.01	PeakCover	Avg	0.8220	0.8458	0.8806	0.5977	0.6601	0.2218	0.7601
			StdDev	0.2607	0.2483	0.2388	0.2444	0.2775	0.1410	0.3061
		PeakDist	Avg	0.2489	0.3555	0.3537	0.3572	0.1915	0.2089	0.5262
			StdDev	0.2443	0.1937	0.2143	0.2154	0.2303	0.2887	0.1786
	0.05	PeakCover	Avg	0.6830	0.6809	0.6847	0.4423	0.5480	0.2086	0.5062
			StdDev	0.2500	0.2916	0.2952	0.2627	0.3121	0.1454	0.3044
		PeakDist	Avg	0.2221	0.4677	0.4182	0.4652	0.2910	0.2667	0.5952
			StdDev	0.2396	0.2108	0.2320	0.2591	0.2849	0.3265	0.2120
	0.1	PeakCover	Avg	0.6338	0.5835	0.5861	0.3905	0.5208	0.2108	0.4278
			StdDev	0.2635	0.3061	0.2732	0.2647	0.3362	0.1859	0.2801
		PeakDist	Avg	0.2428	0.5385	0.3389	0.5072	0.3379	0.3001	0.6055
			StdDev	0.2596	0.2106	0.2702	0.2664	0.3002	0.3667	0.2206
	0.3	PeakCover	Avg	0.6073	0.5227	0.5341	0.3599	0.4890	0.2134	0.4090
			StdDev	0.2615	0.2962	0.2754	0.2668	0.3228	0.1803	0.2847
		PeakDist	Avg	0.2546	0.5581	0.3750	0.5355	0.3367	0.2911	0.6295
			StdDev	0.2657	0.2150	0.2731	0.2739	0.3032	0.3558	0.2174

Chapter 7

Conclusion and Future Work

This final chapter provides a summarized overview of the observations and conclusions made in this work. Suggested future work for niching algorithms, performance metrics, and benchmarks is also discussed.

7.1 Summary and Conclusions

Chapter 3 provided a landscape characterization of the MPB and the GDBG benchmark functions was conducted. Results revealed that the change operators of the benchmark generators do not significantly affect the landscape features. A general classification of the different functions based on ruggedness, searchability, and gradients was provided in Table 3.8 for reference.

Chapter 6 provided experimentation on the niching performance of algorithms reviewed in Chapter 5. Results confirm a trade-off in niching performance between the number of optima located and the accuracy to which those optima are exploited. The statistical comparison showed that none of the seven algorithms dominate all others in peak coverage and accuracy simultaneously. CPSOR and ANPSO show best overall coverage performance when optimized for coverage; AMSO showed to scale slightly better than other algorithms in terms of coverage in abruptly changing environments. In terms of niching accuracy, the AMSO algorithm dominated the others at the expense of coverage.

Chapter 6 also includes a correlation study between niching performance and the landscape features measured in Chapter 3. The results show varying levels of effects depending on the algorithm in question. Overall, gradient measures share a significant

positive correlation with coverage for six of the seven algorithms tested, suggesting that current niching techniques might favour higher gradients when searching for multiple optima. Another correlation was observed between general searchability and the accuracy achieved on multiple optima. This correlation, although slightly lower than the gradients correlation, shows the effect that searchability has on the PSO algorithm used by the individual niches.

7.2 Future Work

This final section suggests three potential research directions based on the findings of this work.

7.2.1 Dynamic Benchmarks

Further research can be done on validating the representativeness of the current benchmark functions. Based on the results obtained in Chapters 3 and 6, the current benchmarks do not vary the landscape features which are shown to have significant correlation with niching performance. Future work could include the landscape feature-measurement of various dynamic, real-world problems to test the regularity in these landscape features. These experiments can also lead to the development of new benchmark functions and/or change operators.

7.2.2 Multi-Modal Performance Metrics

The performance experiments conducted in this work are based on optimization functions where the positions and fitness values of the optima are known in advance. This approach is not feasible in real-world problems as the information on optima is not known. More work is needed in the development of niching performance evaluation that is capable of distinguishing local optima found by an algorithm without prior knowledge given regarding their modality, position, and distribution.

7.2.3 Adaptive Algorithms

A majority of the multi-modal algorithms in this study showed the ability to be adjusted for either coverage or accuracy, with a trade-off in one when optimized for the other. Algorithms such as ANPSO are developed in order to reduce the number of variables that need to be set for a given problem. The way in which this is achieved

is to use information available during optimization to set behavioural parameters in order to adapt behaviour to the problem features. Future development of niching algorithms could make use of niche information to adjust its behaviour for accuracy vs. coverage. This could allow the algorithm to adaptively decide on which goal to focus on at a given state in time, based on current performance. Adaptive behaviour can also be achieved by using the landscape metrics; although new and more efficient ways of determining landscape features are needed for an algorithm to do so efficiently.

Bibliography

[1] Blackwell, Tim, and Jrgen Branke. "Multi-swarm optimization in dynamic environments." Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2004. 489-500.

[2] Blackwell, Tim, and Jrgen Branke. "Multiswarms, exclusion, and anti-convergence in dynamic environments." Evolutionary Computation, IEEE Transactions on 10.4 (2006): 459-472.

[3] Blackwell, Tim, Jrgen Branke, and Xiaodong Li. "Particle swarms for dynamic optimization problems." Swarm Intelligence. Springer Berlin Heidelberg, 2008. 193-217.

[4] Blackwell, Tim. "Particle swarm optimization in dynamic environments." Evolutionary computation in dynamic and uncertain environments. Springer Berlin Heidelberg, 2007. 29-49.

[5] J. Branke. Evolutionary Approaches to Dynamic Optimization Problems-A Survey. GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pages 134-137, 1999.

[6] J. Branke. Memory enhanced evolutionary algorithms for changing optimization-problems. Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, 3, 1999.

[7] J. Branke. Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, 2002.

[8] T.M. Blackwell and P.J. Bentley. Dynamic search with charged swarms. In: Proceedings of the genetic and evolutionary computation conference. Citeseer.2002, pp. 19-26.

[9] Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1942-1948 (1995)

[10] M. Clerc and J. Kennedy, The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space, IEEE Trans. Evol.Comput., vol. 6, no. 1, pp. 58-73, Feb. 2002.

[11] RC Eberhart and Y. Shi. Comparing inertia weight and constriction factors in particle swarm optimization [A] Proceedings of the IEEE congress on Evolutionary Computation [C], 2000.

[12] Eberhart, Russell C., and Yuhui Shi. "Comparison between genetic algorithms and particle swarm optimization." Evolutionary Programming VII. Springer Berlin Heidelberg, 1998.

[13] Hu, X. and Eberhart, R.C.: Adaptive particle swarm optimisation: detection and response to dynamic systems. Proc Congress on Evolutionary Computation (2002) 1666-1670.

[14] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC), pages 1671-1676, 2002.

[15] Parrott, Daniel, and Xiaodong Li. "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation." Evolutionary Computation, 2004. CEC2004. Congress on. Vol. 1. IEEE, 2004.

[16] Parrott, Daniel, and Xiaodong Li. "Locating and tracking multiple dynamic optima by a particle swarm model using speciation." Evolutionary Computation, IEEE Transactions on 10.4 (2006): 440-458.

[17] Bird, Stefan, and Xiaodong Li. "Adaptively choosing niching parameters in a PSO." Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, 2006.

[18] Li, Changhe, Shengxiang Yang, and Ming Yang. "An Adaptive Multi-Swarm Optimizer for Dynamic Optimization Problems." (2014).

[19] Li, Changhe, and Shengxiang Yang. "A general framework of multi population methods with clustering in undetectable dynamic environments." Evolutionary Computation, IEEE Transactions on 16.4 (2012): 556-577.

[20] Li, Changhe, Shengxiang Yang, and Trung Thanh Nguyen. "A self-learning particle swarm optimizer for global optimization problems." Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 42.3 (2012): 627-646.

[21] ang, Shengxiang, and Changhe Li. "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments." Evolutionary Computation, IEEE Transactions on 14.6 (2010): 959-974.

[22] Li, Changhe, and Shengxiang Yang. "A Comparative Study on Particle Swarm Optimization in Dynamic Environments." Evolutionary Computation for Dynamic Optimization Problems. Springer Berlin Heidelberg, 2013. 109-136.

[23] Liang, J. J., P. N. Suganthan, and K. Deb. "Novel composition test functions for numerical global optimization." Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE. IEEE, 2005.

[24] Li, Changhe, and Shengxiang Yang. "A generalized approach to construct benchmark problems for dynamic optimization." Simulated Evolution and Learning. Springer Berlin Heidelberg, 2008. 391-400.

[25] Li, Changhe, Shengxiang Yang, and David Alejandro Pelta. "Benchmark generator for the IEEE WCCI-2012 competition on evolutionary computation for dynamic optimization problems." China University of Geosciences, Brunel University, University of Granada, Tech. Rep (2011).

[26] J. J. Liang, P. N. Suganthan, and K. Deb. Novel composition test functions for numerical global optimization, Proc. of the 2005 IEEE Congr. on Evol. Comput., pp. 68-75, 2005.

[27] Salomon, Ralf. "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms." BioSystems 39.3 (1996): 263-278.

[28] A.P. Engelbrecht. Fundamentals of Computational Swarm Intelligence. Wiley, 2005.

[29] Engelbrecht, A. P. "Particle Swarm Optimization: Global Best or Local Best?." Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS Congress on. IEEE, 2013.

[30] Brits, Riaan, Andries P. Engelbrecht, and F. Van Den Bergh. "Scalability of niche PSO." Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE. IEEE, 2003.

[31] G. Pampara, A. Engelbrecht, and T. Cloete, Cilib: A collaborative framework for computational intelligence algorithms - part i in Proc. of World Congress on Computational Intelligence, Hong Kong, 1-8 June 2008, pp. 1750-1757, source code available at: http://www.cilib.net.

[32] Schoeman, Isabella Lodewina. Niching in particle swarm optimization. Ph.D. dissertation, University of Pretoria, 2010.

[33] Duhain, Julien Georges Omer Louis. "Particle swarm optimisation in dynamically changing environments-an empirical study.", Masters thesis, University of Pretoria, Pretoria, South Africa, 2012.

[34] RC Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, 1, 2001.

[35] X. Hu and RC Eberhart. Tracking dynamic systems with PSO: wheres the cheese. Proceedings of the Workshop on Particle Swarm Optimization, 2001.

[36] P.J. Angeline. Tracking extrema in dynamic environments. Proceedings of the Sixth International Conference on Evolutionary Programming, EP97, 1213:335-345.

[37] T. Back. On the behavior of evolutionary algorithms in dynamic environments. In The 1998 IEEE International Conference on Evolutionary Computation, ICEC98, pages 446-451, 1998.

[38] K. De Jong. Evolving in a changing world. Lecture notes in computer science, pages 512-519, 1999.

[39] Weicker, Karsten. "Performance measures for dynamic environments." Parallel Problem Solving from NaturePPSN VII. Springer Berlin Heidelberg, 2002. 64-73.

[40] A. M. Sutton, D. Whitley, M. Lunacek, and A. Howe. PSO and multi-funnel landscapes: how cooperation might limit exploration. In Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference, pages 75-82, 2006.

[41] Malan, Katherine M., and Andries P. Engelbrecht. "A survey of techniques for characterising fitness landscapes and some possible ways forward." Information Sciences 241 (2013): 148-163.

[42] Malan, Katherine Mary. Characterising Continuous Optimisation Problems for Particle Swarm Optimisation Performance Prediction. Ph.D. dissertation, University of Pretoria, 2014.

[43] E. Izquierdo-Torres. Evolving Dynamical Systems: Nearly Neutral Regions in Continuous Fitness Landscapes. Masters thesis, University of Sussex, 2004.

[44] F. Van den Bergh. An Analysis of Particle Swarm Optimizers. PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2001.

[45] V. K. Vassilev. Fitness Landscapes and Search in the Evolutionary Design of Digital Circuits. Phd thesis, Napier University, 2000.

[46] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information Characteristics and the Structure of Landscapes. Evolutionary Computation, 8(1):31-60, 2000.

[47] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Smoothness, Ruggedness and Neutrality of Fitness Landscapes: from Theory to Application. In Advances in Evolutionary Computing: Theory and Applications, pages 3-44. Springer-Verlag New York, Inc., 2003.

[48] Turney, Peter. Increasing evolvability considered as a large-scale trend in evolution. In Wu, A., editor, Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program (GECCO-99 Workshop on Evolvability), pages 43-46, Morgan Kaufmann, San Mateo, California., 1999.

[49] Reidys, Christian M., and Peter F. Stadler. "Neutrality in fitness landscapes." Applied Mathematics and Computation 117.2 (2001): 321-350.

[50] M. Lunacek and D. Whitley. The dispersion metric and the CMA evolution strategy. In Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference, pages 477-484, 2006.

[51] T. Jones and S. Forrest. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In Proceedings of the Sixth International Conference on Genetic Algorithms, pages 184-192, 1995.

[52] S. Wright. The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in evolution. In Proceedings of the Sixth International Congress on Genetics, pages 356-366, 1932.

[53] T. Jones. Evolutionary Algorithms, Fitness Landscapes and Search. Phd thesis, The University of New Mexico, 1995.

[54] P. F. Stadler. Fitness Landscapes. In A. V. Michael Lassig, editor, Biological Evolution and Statistical Physics, volume 585 of Lecture Notes in Physics, pages 183-204. Springer-Verlag, Berlin, Heidelberg, 2002.

[55] J. Horn and D. E. Goldberg. Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. In L. D. Whitley and M. D. Vose, editors, Foundations of Genetic Algorithms 3, pages 243-269. Morgan Kaufmann, San Francisco, CA, 1995.

[56] Birattari, Mauro, et al. "F-Race and iterated F-Race: An overview." Experimental methods for the analysis of optimization algorithms. Springer Berlin Heidelberg, 2010. 311-336.

[57] M Birattari, T. Stutzle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editors, GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 11-18. Morgan Kaufmann Publishers, San Francisco, CA, 2002.

[58] D. Beasley, D. R. Bull, and R. R. Martin, A Sequential Niching Technique for Multimodal Function Optimization, Evolutionary Computation, vol. 1, no. 2, pp. 101 - 125, 1993.

[59] Carlisle, Anthony Jack, and G. Dozier. Applying the particle swarm optimizer to non-stationary environments. Ph.D. dissertation, Auburn, Alabama: Auburn University, 2002.

[60] Marde Helbig and Andries P. Engelbrecht. Analysing the Performance of Dynamic Multi-objective Optimisation Algorithms. In 2013 IEEE Congress on Evolutionary Computation (CEC2013), pages 1531-1539, Cancun, Mexico, 20-23 June 2013. IEEE Press. ISBN 978-1-4799-0454-9.

Appendix A

Benchmark Landscape Metric Results

This appendix contains the landscape characterization metric results for each of the benchmark functions discussed in this paper.

Table A.1: The average metric readings from 30 generated environments under each generator configuration, (no changes, 1 landscape per environment)

		peaks Avg	5 StDev	peaks Avg	10 StDev	peaks Avg	50 StDev
MPB							
R(0.0:100.0)5̂	FEMmicro	0.2470139535	0.0447417694	0.2537730661	0.0353070641	0.2626615675	0.0253277993
	FEMmacro	0.3092531749	0.0232058693	0.3133766163	0.0128507525	0.3724047816	0.0500456596
	DM	-0.2007864096	0.0783315773	-0.1626953061	0.061908861	-0.0844137612	0.0502962597
	GMavg	4.32593281136	1.3222518451	4.323716218	1.6108960368	5.2875154509	1.2355083235
	GMstd	4.7182089019	0.9280002893	5.3212635609	1.5189730226	6.6843497604	1.7767375919
	FDC	0.4764177941	0.1484577652	0.4193758834	0.1799086194	0.2432581404	0.1164596972
R(0.0:100.0)1̂0	FEMmicro	0.2276347077	0.0264925754	0.2270481886	0.0377818104	0.2352326338	0.0330631051
	FEMmacro	0.3068958622	0.0098469694	0.3141100121	0.0137679265	0.3197180142	0.0105919283
	DM	-0.2252574534	0.0457183672	-0.2095003066	0.0462717175	-0.1698389124	0.0443655203
	GMavg	4.3597313444	1.2464206185	5.6057462114	1.5572555823	5.6435689286	1.9086550501
	GMstd	4.289693608	0.807183975	4.8710399129	1.012857472	5.6587539562	1.4285082053
	FDC	0.5757841232	0.1802855615	0.5826746025	0.1718910012	0.4339763646	0.1744885256
R(0.0:100.0)3̂0	FEMmicro	0.1758008506	0.0272788444	0.1822121707	0.0317867549	0.1884487802	0.0314837773
	FEMmacro	0.3097993574	0.0090014106	0.3140289107	0.007891022	0.3130583802	0.0061532101
	DM	-0.2496283816	0.0136396512	-0.2471481794	0.0206805648	-0.2340459958	0.0219348013
	GMavg	5.6542167005	0.9643824967	5.6926482284	1.2729374524	5.7375650185	1.2755789678
	GMstd	4.8054075908	0.573544519	4.8050491095	0.6570410911	4.8977384339	0.797190165
	FDC	0.6145594309	0.0876116409	0.5552972082	0.1245514079	0.5436187213	0.1082029957
ackley							
R(-5.0:5.0)5̂	FEMmicro	0.8667636678	0.0083869071	0.8667710447	0.0100715558	0.8646992961	0.0068925276
	FEMmacro	0.7995122208	0.0373946387	0.7693185983	0.0495108789	0.7388767185	0.0248615317
	DM	0.0602725475	0.0410476042	0.0681675513	0.048656961	0.1356334058	0.0402736329
	GMavg	53.9392667145	22.470666651	47.4278277461	17.0521846203	31.3405134844	8.9790391583
	GMstd	39.8465902787	16.5225240018	35.0746529435	12.1503597012	23.672496066	6.6505017878
	FDC	-0.1337862669	0.0929559241	-0.1323349543	0.1215662547	-0.0405318239	0.097389761
R(-5.0:5.0)1̂0	FEMmicro	0.8668019341	0.0057346627	0.86994278	0.00614787	0.8672334506	0.0062058876
	FEMmacro	0.8544822165	0.0108777697	0.8464187831	0.0112502875	0.8166865219	0.0117256786
	DM	0.0704351359	0.0257889578	0.0876914742	0.028218879	0.1805941726	0.0224511802
	GMavg	90.1654861288	17.7041934591	78.9162055709	15.9341880524	51.4111952246	9.1703130542
	GMstd	66.4384309845	13.5406263424	58.4651718675	11.7981677435	38.6113787617	7.0470228302
	FDC	-0.0541061562	0.0908305429	-0.0532722468	0.1003577844	0.0787205577	0.0892831853
R(-5.0:5.0)3̂0	FEMmicro	0.868451791	0.00455028	0.8702720072	0.0053925513	0.8705212169	0.0051052114
	FEMmacro	0.8656741979	0.0051131534	0.8651102474	0.0049832542	0.855195636	0.00593794
	DM	0.0312429624	0.0146650172	0.0630861486	0.0106299913	0.1505632064	0.0103306367
	GMavg	98.8945827881	16.8256559475	81.6739775341	10.538433138	51.226700682	6.4933053924
	GMstd	73.2756436628	12.4317355981	61.1041298533	8.0874796947	38.5605001802	4.7514084804
	FDC	-0.0015100213	0.0708869929	-0.0043544785	0.0713244004	0.1672172423	0.0657120735
griewank							
R(-5.0:5.0)5̂	FEMmicro	0.7965938154	0.0258458026	0.7821458132	0.0257278907	0.7178103357	0.0419924502
	FEMmacro	0.6449314436	0.0305112102	0.6315207789	0.0309695206	0.6541094367	0.0239294237
	DM	0.0555217238	0.0623529302	0.0747277019	0.0742926143	0.2054746079	0.0345432953
	GMavg	8.3252546668	2.952326655	7.533915447	2.537979523	5.6848239695	1.5257519673
	GMstd	7.6423019038	2.8709718493	6.4434256179	2.3963444026	4.3133136997	1.1834480364
	FDC	-0.3338301873	0.1651593021	-0.2468771683	0.1864981142	0.0069504114	0.1280936195
R(-5.0:5.0)1̂0	FEMmicro	0.4965050827	0.0710264782	0.4687442256	0.0767482691	0.4322656044	0.055408536
	FEMmacro	0.6360354212	0.0135893225	0.6414736469	0.0199863834	0.6532074359	0.0197320879
	DM	0.0943092511	0.040888135	0.1384375944	0.0396730872	0.2282051359	0.0240699401
	GMavg	4.1853149989	1.4136502482	4.4523208323	1.085135745	4.3081168013	1.0568029011
	GMstd	2.9006972845	0.9898094374	3.0891279035	0.7428348507	2.7138889899	0.663554743
	FDC	-0.2287300554	0.1234323504	-0.1331608549	0.1254040536	0.0907029796	0.1017850621
R(-5.0:5.0)3̂0	FEMmicro	0.3519478077	0.0318869604	0.3367135829	0.0287189446	0.3409517339	0.0268814012
	FEMmacro	0.6409430934	0.0168382973	0.6483437563	0.013213005	0.6553396854	0.0133079956
	DM	0.1086428507	0.0181313299	0.1574627441	0.0191563107	0.2229291685	0.0093405775
	GMavg	4.610227807	0.6905849006	4.5147337631	0.5286504288	4.2469366749	0.4856101617
	GMstd	3.1195650681	0.5143120231	2.8463826214	0.3493483042	2.5666563665	0.2929459724
	FDC	-0.0652736261	0.0608044327	0.0660344978	0.0781852974	0.2243838712	0.0490614793
rastrigin							
R(-5.0:5.0)5̂	FEMmicro	0.5997645285	0.0192399099	0.5870155158	0.0197801646	0.5757049148	0.0218796019
	FEMmacro	0.8447863754	0.0124763008	0.8385062779	0.0133177528	0.7986188586	0.0179810264
	DM	0.0633476039	0.0662238948	0.0947737365	0.0681063666	0.1843856397	0.04992441
	GMavg	11.5192174417	4.1421935464	11.2596615601	3.2783337583	8.0625202958	2.9669123317
	GMstd	8.0473562598	2.7208390205	8.0891154893	2.2894855963	5.9869555269	2.0158393258
	FDC	-0.295089833	0.1880118474	-0.2155263822	0.1861639362	-0.0015685542	0.149180251
R(-5.0:5.0)1̂0	FEMmicro	0.6009612562	0.0176896268	0.5846851895	0.0184852767	0.5461743355	0.030777424
	FEMmacro	0.840345878	0.0109952677	0.8221245096	0.0093580561	0.7556518621	0.0189059485
	DM	0.0921459667	0.0340367459	0.1381811861	0.031837037	0.2321499681	0.0222313082
	GMavg	12.7093990594	3.2047834399	11.494335493	2.9859490035	6.743198077	1.2927266846
	GMstd	9.2088158769	2.3110356814	8.4067914109	2.1842418472	5.0831645508	1.0112401695
	FDC	-0.2010298987	0.1409255967	-0.1287358018	0.0938043169	0.1344594454	0.0936843058
R(-5.0:5.0)3̂0	FEMmicro	0.5736772934	0.0167100289	0.5433857771	0.0252719869	0.446931044	0.0326348261
	FEMmacro	0.8350464555	0.0048913957	0.8089283572	0.0097974953	0.7296959875	0.0140236871
	DM	0.102999379	0.0160253611	0.1467681997	0.0187472181	0.2215583258	0.0098249721
	GMavg	13.2181230678	2.3834483445	10.5831813002	1.6353097323	6.3887044231	0.8441155582
	GMstd	9.7306300639	1.7928851987	7.9417790298	1.2052527276	4.6976172839	0.6254892595
	FDC	-0.0280448674	0.0698733477	0.0742064933	0.0749663468	0.2149024296	0.0410905156
sphere							
R(-5.0:5.0)5̂	FEMmicro	0.5029395931	0.069971271	0.4941437226	0.0599675342	0.4616756735	0.0660788122
	FEMmacro	0.6164925755	0.0224107476	0.6229509737	0.0257853347	0.6395443665	0.0310104919
	DM	0.0447497036	0.0641118927	0.0987105152	0.0589691318	0.1901033944	0.0478696348
	GMavg	3.6230171962	0.9568781695	4.1820640175	1.1477537445	3.6236455771	1.1331583575
	GMstd	2.4922736859	0.524940628	2.9460806796	0.8465373932	2.5797003751	0.7536266328
	FDC	-0.3533140604	0.148654852	-0.2453832641	0.135583085	-0.0217422906	0.1537988642
R(-5.0:5.0)1̂0	FEMmicro	0.4426182035	0.0622644512	0.4339249025	0.0552277627	0.4608575505	0.0703294355
	FEMmacro	0.6333977946	0.0146082802	0.6357888831	0.0195356996	0.6523871564	0.0160939758
	DM	0.0934909203	0.0333902274	0.1414946744	0.0391288218	0.2361675034	0.022285351
	GMavg	3.8435806387	0.8359771055	4.2855337339	0.8682926526	4.0837302368	0.7399280734
	GMstd	2.624157471	0.5973743182	2.8681966973	0.6041826567	2.5420928743	0.4138215897
	FDC	-0.2392154295	0.1301209777	-0.1387483731	0.111651513	0.1276348116	0.0801959206
R(-5.0:5.0)3̂0	FEMmicro	0.3546781953	0.0389253614	0.3402802013	0.0338256211	0.3340624445	0.0238369627
	FEMmacro	0.6479666097	0.0151424065	0.6447633692	0.0161762726	0.6501822243	0.0147326445
	DM	0.1062797137	0.0158423002	0.1554210353	0.0215340989	0.2253756655	0.0090840878
	GMavg	4.5183987843	0.8271351113	4.5148257546	0.6785873833	4.3135131266	0.5583738168
	GMstd	3.0225166306	0.5334213237	2.8800338837	0.4302482009	2.6089292245	0.3363110381
	FDC	-0.0340869694	0.085629088	0.0665856517	0.0750376384	0.2266458284	0.0511825721
weierstrass							
R(-5.0:5.0)5̂	FEMmicro	0.7849304616	0.0144084847	0.7803710872	0.0131368356	0.781388806	0.013779585
	FEMmacro	0.8039617065	0.0127288788	0.8014025518	0.015757742	0.7928845148	0.0130682047
	DM	-0.0363387865	0.0732345229	0.0185134667	0.0646471683	0.0641392965	0.0650455182
	GMavg	17.0142953816	4.1490664208	16.433674765	3.5145046642	15.3774230827	4.9205301264
	GMstd	12.6495856316	3.0350956656	12.3550636997	2.4720701904	11.6013611279	3.5982707846
	FDC	-0.2469383641	0.169256292	-0.1600443162	0.1210100256	-0.1155617113	0.1589529666
R(-5.0:5.0)1̂0	FEMmicro	0.7906084888	0.0087823937	0.7889806384	0.0088772356	0.7840064538	0.0088930821
	FEMmacro	0.8082502022	0.0099977369	0.8066850367	0.0099933726	0.7979750481	0.0082072218
	DM	-0.0029138048	0.0450125506	0.022378296	0.0437706248	0.0740069348	0.0379716071
	GMavg	24.6569786521	5.7981455821	25.4463044606	4.9804255218	21.1494406911	5.1010747483
	GMstd	18.31373120065	4.2306662309	19.0705121401	3.6983199338	16.0221963818	3.8911848405
	FDC	-0.1282161042	0.0991642268	-0.1029131383	0.1070746396	-0.0328403042	0.0925154481
R(-5.0:5.0)3̂0	FEMmicro	0.7904713681	0.0069158325	0.7887799388	0.0076829477	0.7857125411	0.0066551224
	FEMmacro	0.8071579383	0.0061128533	0.8093668866	0.0068295327	0.8034447058	0.0088583814
	DM	-0.0328319023	0.0271612922	-0.0225179088	0.0245456796	0.027175153	0.024099543
	GMavg	31.8190226205	6.5212307864	27.7208428759	5.1597462442	24.3016608312	5.4113793438
	GMstd	23.6777270051	4.8021391639	20.7188461348	3.7324217074	18.3503472829	4.0420841093
	FDC	-0.074894535	0.0608738917	-0.0342898059	0.0688452543	0.0088482061	0.07063417

Table A.2: The average metric readings from 30 generated environments under each generator configuration, (high spatial severity, 10 landscape per environment)

		peaks 5 Avg	peaks 5 STDev	peaks 10 Avg	peaks 10 STDev	peaks 50 Avg	peaks 50 STDev
MPB							
R(0.0:100.0)5̂	FEMmicro	0.2638397019	0.0357791445	0.2609748265	0.0342174869	0.2739832011	0.0278606385
	FEMmacro	0.3110971915	0.0160647526	0.3153743748	0.01371424	0.3437189944	0.0362339652
	DM	-0.2215111667	0.0585378846	-0.2023636077	0.0628032567	-0.1722525912	0.067548475
	GMavg	4.7978186963	1.6683060666	4.8437598046	1.6083001391	5.3772499468	1.8722309101
	GMstd	5.0599759668	1.2877190546	5.3992971714	1.3418913803	6.5712848789	1.8338115254
	FDC	0.4488282426	0.1683629723	0.4130194171	0.1787839919	0.3383593581	0.1685941266
R(0.0:100.0)1̂0	FEMmicro	0.2662061755	0.0321094691	0.2710228276	0.0359100915	0.274781202	0.0326098123
	FEMmacro	0.3130850705	0.0149235884	0.3146930419	0.0173104199	0.3236712083	0.0273957456
	DM	-0.2360964012	0.0373490581	-0.2285208366	0.0433303112	-0.2154056702	0.0473945375
	GMavg	6.1428390663	2.3170122361	6.3003445018	2.3574541535	6.3277886089	2.3625416055
	GMstd	5.2677371325	1.4590428469	5.4494786774	1.5265583474	5.6501221705	1.6233600022
	FDC	0.6135572025	0.1572727709	0.5811823627	0.1755044914	0.5387731286	0.179002476
R(0.0:100.0)3̂0	FEMmicro	0.2558948602	0.0465055152	0.2617279157	0.0447533196	0.2645223857	0.0425291798
	FEMmacro	0.3124535501	0.0163851384	0.3118990358	0.0139491031	0.3153932793	0.0190556879
	DM	-0.253160059	0.0218589664	-0.2535481728	0.0230269997	-0.2484573101	0.0235270792
	GMavg	9.2660724782	3.3607718376	9.2016427648	3.5972560631	9.3001517123	3.4706308198
	GMstd	7.0353136191	2.2595880283	7.001313631	2.4158565666	7.1183538065	2.3477925835
	FDC	0.5709314824	0.087130301	0.5706734981	0.0931932992	0.5427989896	0.1036887602
ackley							
R(-5.0:5.0)5̂	FEMmicro	0.866582958	0.0090064987	0.866554887	0.0091764625	0.8657518062	0.0092852106
	FEMmacro	0.8020500756	0.0446608079	0.7760201847	0.050268821	0.7439036662	0.0283060872
	DM	0.0437124843	0.0362057956	0.0667096388	0.0428505345	0.144668229	0.0379800996
	GMavg	56.7127109289	18.9042503049	48.2613974462	16.6860513112	32.4191535424	10.2929952615
	GMstd	41.6289508234	13.8526313119	35.6830672957	12.1797738895	24.2104502982	7.6052392186
	FDC	-0.1317279651	0.1125386718	-0.1055173695	0.1017412222	-0.0267300276	0.0990532507
R(-5.0:5.0)1̂0	FEMmicro	0.8678702417	0.0085606383	0.8680696769	0.0089822015	0.8683167077	0.0084642776
	FEMmacro	0.8530862701	0.0122344497	0.8472541926	0.013341933	0.822106619	0.012735475
	DM	0.0608056799	0.023114813	0.0891737186	0.0261686238	0.1734725599	0.0243577927
	GMavg	70.086554202	15.3915733305	61.9785846887	15.07004102	48.1328926595	12.0247600789
	GMstd	52.2362703802	11.5063532479	46.5061146549	11.2600050146	36.2448951512	9.0442523486
	FDC	-0.0802968822	0.1030967332	-0.0513926808	0.0964667402	0.0723595464	0.0903452158
R(-5.0:5.0)3̂0	FEMmicro	0.8693266573	0.0081789793	0.869261197	0.0084872153	0.8692216841	0.0083813052
	FEMmacro	0.8668609933	0.0081590577	0.8651438907	0.0088023456	0.8558149583	0.0093357109
	DM	0.0311314637	0.0189880493	0.059332511	0.0183607461	0.1475541481	0.0172753246
	GMavg	42.7196663837	6.5515947579	39.5042426385	6.7876614134	32.4579609421	7.1331805054
	GMstd	31.7385995209	4.9062215735	29.5868302936	5.1042689255	24.4283410627	5.3753722472
	FDC	-0.0155508613	0.0699644542	0.0159597816	0.0722451468	0.138502463	0.0606544753
griewank							
R(-5.0:5.0)5̂	FEMmicro	0.8098270061	0.0208338315	0.8027599636	0.0277808771	0.7429569756	0.0451577788
	FEMmacro	0.6437758238	0.0244713875	0.6462911627	0.0257702483	0.6465577978	0.0254971193
	DM	0.0303784011	0.0516185238	0.0800470074	0.0536056107	0.2076082834	0.0354536651
	GMavg	9.2456674408	3.476638207	8.785955951	3.3384419158	7.2500279586	2.3580662571
	GMstd	8.3842943942	3.3247281298	7.4989854339	3.0126582299	5.5322567865	1.815141179
	FDC	-0.3598545784	0.139863448	-0.2441582611	0.1357241053	0.0167871674	0.1245294158
R(-5.0:5.0)1̂0	FEMmicro	0.5736057138	0.073165633	0.5594600608	0.0737716544	0.5463954382	0.0840300307
	FEMmacro	0.6395594549	0.0247109886	0.6447138563	0.0230648697	0.6535469998	0.0252713378
	DM	0.0976038049	0.039635772	0.1401426629	0.0336875977	0.2358861223	0.02414267
	GMavg	6.1355222276	2.3275365274	6.6253041304	2.4184575226	7.4857828014	2.758032438
	GMstd	4.2005141012	1.6014058556	4.3948016892	1.6182117426	4.5705096224	1.6622624554
	FDC	-0.2122391379	0.1198561276	-0.1073328483	0.1197112088	0.1180784006	0.0937502183
R(-5.0:5.0)3̂0	FEMmicro	0.5215429537	0.0926669435	0.5349687577	0.0945966785	0.5283295399	0.097331593
	FEMmacro	0.6489840533	0.0244156547	0.6535394038	0.0238352573	0.6600860479	0.0243376047
	DM	0.1072867526	0.0239760243	0.1532682	0.0219914622	0.2261406869	0.0148169871
	GMavg	10.5655834982	3.7995751174	10.6024173132	3.6300489141	11.608654532	3.8320943015
	GMstd	6.9189723039	2.4852277311	6.6395089581	2.2647465983	6.8480324311	2.2446696128
	FDC	-0.0561731287	0.0767573815	0.0440706429	0.0721082581	0.2060310693	0.0518733646
rastrigin							
R(-5.0:5.0)5̂	FEMmicro	0.5955862538	0.0188032013	0.5937077979	0.0185908109	0.5877668201	0.022667222
	FEMmacro	0.8476502994	0.0108262986	0.8398326284	0.0118044096	0.7989558873	0.0159568834
	DM	0.048706165	0.0503811042	0.0791934403	0.0569029302	0.2012442734	0.0342261271
	GMavg	13.3886199123	3.4616781402	12.2951463286	3.3363820107	9.5102327448	2.7777116092
	GMstd	9.3213539029	2.3875433164	8.8883981096	2.3592507932	7.0677127886	2.0578760187
	FDC	-0.3113705366	0.1353529002	-0.2413260035	0.1513835108	0.0080282324	0.1147338883
R(-5.0:5.0)1̂0	FEMmicro	0.6044599186	0.0200449184	0.6016294189	0.0201130964	0.5908518508	0.0313096405
	FEMmacro	0.8409930185	0.0106334159	0.8254869175	0.0126775387	0.7583445748	0.0189296388
	DM	0.0946745661	0.0353301578	0.1418005138	0.0335187054	0.2357551077	0.0210768541
	GMavg	15.9109431726	4.2729823461	15.040248662	4.1762124516	10.4192306725	3.3480431252
	GMstd	11.4131898973	3.0176487539	11.0675660038	3.0466086969	7.7317741461	2.4885066503
	FDC	-0.1913693499	0.1193417361	-0.0941804052	0.1128192032	0.1138156438	0.0888890088
R(-5.0:5.0)3̂0	FEMmicro	0.6055042422	0.0222679755	0.6012215036	0.0281005528	0.5703637199	0.060803079
	FEMmacro	0.8344890695	0.0116961324	0.8126514612	0.0142937521	0.7359473131	0.0202398128
	DM	0.1032616954	0.0228571847	0.1485382144	0.0215738844	0.2239719203	0.0153869067
	GMavg	23.2589909504	5.9789655743	20.6473393887	5.5173353173	15.023694899	4.7063054703
	GMstd	17.0375029902	4.3833713779	15.3542883	4.1000672403	10.9549761374	3.4499103954
	FDC	-0.0451436075	0.0775260656	0.0498175789	0.0745256423	0.2028177301	0.0520015663
sphere							
R(-5.0:5.0)5̂	FEMmicro	0.5393030017	0.0626012472	0.5399163833	0.0676153931	0.5413918702	0.0754647925
	FEMmacro	0.6197960716	0.0253464072	0.626360705	0.0254345277	0.638462349	0.026187075
	DM	0.0337998461	0.0517608688	0.0821594603	0.0596689869	0.2002597053	0.0388652758
	GMavg	4.1577884388	1.5022098735	4.9612705479	1.9029481688	5.1266510653	1.7417765239
	GMstd	2.8460913219	1.0196313828	3.3319482071	1.2101286652	3.3955887711	1.0784732332
	FDC	-0.3604016406	0.1373797934	-0.2382913283	0.1507330559	-0.0054462781	0.121263542
R(-5.0:5.0)1̂0	FEMmicro	0.550971302	0.0718374917	0.5414776116	0.0726631789	0.5475643315	0.0821320288
	FEMmacro	0.6381165236	0.0246647664	0.6430970178	0.0243230999	0.6510419874	0.0248307282
	DM	0.0901278035	0.0353930188	0.136583222	0.0367156837	0.2384699441	0.0236134526
	GMavg	6.2305191695	2.3216944787	6.5745265812	2.3986627678	7.030584642	2.4578627584
	GMstd	4.2083317644	1.5803147103	4.2755448417	1.5643276902	4.2726901194	1.4599780809
	FDC	-0.2211378838	0.1165016921	-0.1152606569	0.1240772223	0.1210405033	0.0935193275
R(-5.0:5.0)3̂0	FEMmicro	0.5263883665	0.0955963312	0.5249438367	0.0877486651	0.5256412198	0.0946026763
	FEMmacro	0.6505949299	0.0234737971	0.6545109051	0.0230226411	0.6591162751	0.0238941646
	DM	0.1069679063	0.0240948681	0.1519382538	0.0209087449	0.2261147713	0.0144976034
	GMavg	10.3895588735	3.7023520654	10.8190532354	3.7976524837	11.6283594497	4.0936567668
	GMstd	6.8205007637	2.4224885196	6.7733885106	2.3761982261	6.8619034176	2.4002886579
	FDC	-0.0531128927	0.0784387086	0.0431950623	0.0711261813	0.2036822463	0.0532701715
weierstrass							
R(-5.0:5.0)5̂	FEMmicro	0.7872900898	0.0118736078	0.7856120471	0.012137706	0.7829844105	0.0132007065
	FEMmacro	0.803337222	0.0116900751	0.7972867613	0.0122497124	0.7865149782	0.0130663218
	DM	0.0116012581	0.0475377628	0.033433207	0.0437183474	0.111350552	0.0427550416
	GMavg	16.7860182154	4.0336223929	16.1236717947	3.571762084	14.3636960984	3.4966796308
	GMstd	12.4144475639	2.9708234578	12.0751933421	2.6267298	10.873355427	2.6534995827
	FDC	-0.1728386374	0.1265493738	-0.1258108544	0.1278732447	-0.0329233314	0.123768018
R(-5.0:5.0)1̂0	FEMmicro	0.7892832787	0.0113612871	0.7891294927	0.0115179634	0.7858756622	0.012053389
	FEMmacro	0.806990782	0.0111397385	0.8045834164	0.0111643107	0.7912908147	0.0134101039
	DM	0.0273868877	0.0360578256	0.0464493277	0.0382762548	0.1315688547	0.03660251
	GMavg	18.8869359802	4.6056938468	18.3862313238	4.2799713532	16.6442226322	4.1651534662
	GMstd	13.981825269	3.3993386802	13.7450380404	3.1935248981	12.5542560064	3.1620596132
	FDC	-0.0923386149	0.1074677177	-0.0658927559	0.1057158158	0.0455735906	0.103337214
R(-5.0:5.0)3̂0	FEMmicro	0.7898598865	0.012187363	0.7887438748	0.0113461415	0.7853024113	0.0121056271
	FEMmacro	0.8072353787	0.010324488	0.8061989722	0.0108428893	0.7908479456	0.0144931249
	DM	-0.0012230347	0.0276870383	0.0156453802	0.0327375825	0.09543814	0.0403113241
	GMavg	20.5941895266	4.4349306857	20.6766371823	4.5554636447	18.8622428816	4.7059762647
	GMstd	15.2997758236	3.3009682781	15.4769044924	3.4145673933	14.2050309024	3.5503043641
	FDC	-0.0372486121	0.0700130367	-0.0153246875	0.0712349591	0.0903632391	0.0773924948

Appendix B

Multi-modal PSO Algorithm Comparison

This appendix contains results from the statistical comparison of niching algorithms. The results are summarized in terms of the benchmark function, modality, and shift severities used in the experiments.

Table B.1: Wins and losses for best algorithm configurations selected based on $offline_{Error}$ for each of the benchmark functions

Problem	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Ackley	Error	Wins	54	31	43	11	0	9	22
		Losses	0	18	9	26	54	38	25
		Diff	54	13	34	-15	-54	-29	-3
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	33	52	40	12	32	0	14
		Losses	20	1	12	41	19	54	36
		Diff	13	51	28	-29	13	-54	-22
		Rank	3	**1**	2	6	3	7	5
	PeakDist	Wins	51	18	31	2	37	36	5
		Losses	0	36	20	50	13	14	47
		Diff	51	-18	11	-48	24	22	-42
		Rank	**1**	5	4	7	2	3	6
Griewank	Error	Wins	53	36	40	16	0	9	21
		Losses	1	13	12	26	54	43	26
		Diff	52	23	28	-10	-54	-34	-5
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	33	54	40	9	22	0	26
		Losses	18	0	13	44	29	54	26
		Diff	15	54	27	-35	-7	-54	0
		Rank	3	**1**	2	6	5	7	4
	PeakDist	Wins	51	29	33	3	39	17	7
		Losses	0	22	19	49	9	35	45
		Diff	51	7	14	-46	30	-18	-38
		Rank	**1**	4	3	7	2	5	6
Mpb	Error	Wins	44	40	35	0	0	15	29
		Losses	0	7	9	44	43	36	24
		Diff	44	33	26	-44	-43	-21	5
		Rank	**1**	2	3	7	6	5	4
	PeakCover	Wins	31	41	44	17	29	0	12
		Losses	17	3	4	37	21	53	39
		Diff	14	38	40	-20	8	-53	-27
		Rank	3	2	**1**	5	4	7	6
	PeakDist	Wins	48	20	29	8	42	35	0
		Losses	4	34	24	43	10	14	53
		Diff	44	-14	5	-35	32	21	-53
		Rank	**1**	5	4	6	2	3	7
Rastrigin	Error	Wins	54	30	44	11	0	10	25
		Losses	0	19	9	31	54	38	23
		Diff	54	11	35	-20	-54	-28	2
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	37	48	27	14	34	0	23
		Losses	15	4	22	40	18	54	30
		Diff	22	44	5	-26	16	-54	-7
		Rank	2	**1**	4	6	3	7	5
	PeakDist	Wins	53	18	33	3	35	29	4
		Losses	0	35	13	49	13	17	48
		Diff	53	-17	20	-46	22	12	-44
		Rank	**1**	5	3	7	2	4	6
Sphere	Error	Wins	54	33	44	16	0	9	25
		Losses	0	18	9	31	54	43	26
		Diff	54	15	35	-15	-54	-34	-1
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	36	52	39	12	25	0	18
		Losses	16	2	13	41	25	54	31
		Diff	20	50	26	-29	0	-54	-13
		Rank	3	**1**	2	6	4	7	5
	PeakDist	Wins	50	20	36	3	40	23	4
		Losses	0	31	14	49	5	29	48
		Diff	50	-11	22	-46	35	-6	-44
		Rank	**1**	5	3	7	2	4	6
Weierstrass	Error	Wins	48	23	37	35	0	9	20
		Losses	2	26	12	4	54	45	29
		Diff	46	-3	25	31	-54	-36	-9
		Rank	**1**	4	3	2	7	6	5
	PeakCover	Wins	34	50	38	4	31	4	20
		Losses	17	2	13	48	20	49	32
		Diff	17	48	25	-44	11	-45	-12
		Rank	3	**1**	2	6	4	7	5
	PeakDist	Wins	52	16	31	1	34	27	8
		Losses	0	33	13	53	11	15	44
		Diff	52	-17	18	-52	23	12	-36
		Rank	**1**	5	3	7	2	4	6

Table B.2: Wins and losses for best algorithm configurations selected based on $offline_{PeakCover}$ for each of the benchmark functions

Problem	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Ackley	Error	Wins	49	35	45	14	6	3	30
		Losses	4	17	6	37	45	49	24
		Diff	45	18	39	-23	-39	-46	6
		Rank	**1**	3	2	5	6	7	4
	PeakCover	Wins	41	42	50	9	18	0	25
		Losses	12	11	2	45	34	54	27
		Diff	29	31	48	-36	-16	-54	-2
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	39	9	23	20	42	45	0
		Losses	10	42	30	33	9	1	53
		Diff	29	-33	-7	-13	33	44	-53
		Rank	3	6	4	5	2	**1**	7
Griewank	Error	Wins	53	39	38	11	5	4	31
		Losses	1	14	15	37	44	47	23
		Diff	52	25	23	-26	-39	-43	8
		Rank	**1**	2	3	5	6	7	4
	PeakCover	Wins	40	46	45	11	23	0	21
		Losses	12	6	9	42	31	54	32
		Diff	28	40	36	-31	-8	-54	-11
		Rank	3	**1**	2	6	4	7	5
	PeakDist	Wins	43	22	18	17	43	32	0
		Losses	9	27	31	36	7	12	53
		Diff	34	-5	-13	-19	36	20	-53
		Rank	2	4	5	6	**1**	3	7
Mpb	Error	Wins	52	35	40	3	9	0	27
		Losses	0	12	11	40	37	43	23
		Diff	52	23	29	-37	-28	-43	4
		Rank	**1**	3	2	6	5	7	4
	PeakCover	Wins	40	41	46	9	22	0	24
		Losses	10	8	5	45	30	54	30
		Diff	30	33	41	-36	-8	-54	-6
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	40	10	21	20	40	52	0
		Losses	13	42	29	31	13	1	54
		Diff	27	-32	-8	-11	27	51	-54
		Rank	2	6	4	5	2	**1**	7
Rastrigin	Error	Wins	51	36	41	9	6	5	29
		Losses	0	14	12	38	42	48	23
		Diff	51	22	29	-29	-36	-43	6
		Rank	**1**	3	2	5	6	7	4
	PeakCover	Wins	37	39	50	10	24	0	24
		Losses	14	14	3	43	29	54	27
		Diff	23	25	47	-33	-5	-54	-3
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	41	9	19	20	44	43	3
		Losses	11	43	32	31	6	5	51
		Diff	30	-34	-13	-11	38	38	-48
		Rank	3	6	5	4	**1**	1	7
Sphere	Error	Wins	53	32	43	14	4	2	29
		Losses	0	18	8	36	45	47	23
		Diff	53	14	35	-22	-41	-45	6
		Rank	**1**	3	2	5	6	7	4
	PeakCover	Wins	41	41	51	9	20	0	24
		Losses	12	12	2	45	32	54	29
		Diff	29	29	49	-36	-12	-54	-5
		Rank	2	2	**1**	6	5	7	4
	PeakDist	Wins	40	11	21	20	41	42	1
		Losses	10	41	31	33	6	2	53
		Diff	30	-30	-10	-13	35	40	-52
		Rank	3	6	4	5	2	**1**	7
Weierstrass	Error	Wins	54	31	43	17	6	3	29
		Losses	0	19	9	35	48	50	22
		Diff	54	12	34	-18	-42	-47	7
		Rank	**1**	3	2	5	6	7	4
	PeakCover	Wins	37	40	50	9	21	0	28
		Losses	16	12	3	45	32	54	23
		Diff	21	28	47	-36	-11	-54	5
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	42	9	23	21	41	43	0
		Losses	7	44	31	33	9	2	53
		Diff	35	-35	-8	-12	32	41	-53
		Rank	2	6	4	5	3	**1**	7

Table B.3: Wins and losses for best algorithm configurations selected based on $offline_{PeakDist}$ for each of the benchmark functions

Problem	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
Ackley	Error	Wins	53	33	41	5	1	11	27
		Losses	0	14	8	40	49	36	24
		Diff	53	19	33	-35	-48	-25	3
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	13	54	25	20	40	0	31
		Losses	39	0	25	31	13	54	21
		Diff	-26	54	0	-11	27	-54	10
		Rank	6	**1**	4	5	2	7	3
	PeakDist	Wins	50	16	33	8	27	33	0
		Losses	0	34	12	43	19	6	53
		Diff	50	-18	21	-35	8	27	-53
		Rank	**1**	5	3	6	4	2	7
Griewank	Error	Wins	54	37	39	1	2	12	27
		Losses	0	12	10	44	43	36	27
		Diff	54	25	29	-43	-41	-24	0
		Rank	**1**	3	2	7	6	5	4
	PeakCover	Wins	15	47	28	22	40	0	26
		Losses	37	5	22	28	10	54	22
		Diff	-22	42	6	-6	30	-54	4
		Rank	6	**1**	3	5	2	7	4
	PeakDist	Wins	53	24	32	9	32	22	0
		Losses	0	22	15	43	15	23	54
		Diff	53	2	17	-34	17	-1	-54
		Rank	**1**	4	2	6	2	5	7
Mpb	Error	Wins	37	47	37	2	8	8	28
		Losses	9	2	10	45	40	41	20
		Diff	28	45	27	-43	-32	-33	8
		Rank	2	**1**	3	7	5	6	4
	PeakCover	Wins	27	52	31	3	26	2	28
		Losses	20	0	19	46	20	47	17
		Diff	7	52	12	-43	6	-45	11
		Rank	4	**1**	2	6	5	7	3
	PeakDist	Wins	40	9	27	18	38	49	0
		Losses	10	45	26	36	10	0	54
		Diff	30	-36	1	-18	28	49	-54
		Rank	2	6	4	5	3	**1**	7
Rastrigin	Error	Wins	54	33	44	2	2	14	30
		Losses	0	20	9	43	45	38	24
		Diff	54	13	35	-41	-43	-24	6
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	14	52	25	22	41	0	28
		Losses	37	2	26	29	11	54	23
		Diff	-23	50	-1	-7	30	-54	5
		Rank	6	**1**	4	5	2	7	3
	PeakDist	Wins	49	14	36	11	29	35	0
		Losses	0	38	13	41	21	7	54
		Diff	49	-24	23	-30	8	28	-54
		Rank	**1**	5	3	6	4	2	7
Sphere	Error	Wins	46	36	45	2	1	11	27
		Losses	2	13	4	42	43	37	27
		Diff	44	23	41	-40	-42	-26	0
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	11	54	33	13	39	0	27
		Losses	38	0	18	36	11	54	20
		Diff	-27	54	15	-23	28	-54	7
		Rank	6	**1**	3	5	2	7	4
	PeakDist	Wins	53	15	35	10	27	26	0
		Losses	0	36	9	42	13	12	54
		Diff	53	-21	26	-32	14	14	-54
		Rank	**1**	5	2	6	3	3	7
Weierstrass	Error	Wins	53	33	43	9	0	9	29
		Losses	0	18	8	36	53	37	24
		Diff	53	15	35	-27	-53	-28	5
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	15	53	23	19	39	0	31
		Losses	38	0	28	31	11	54	18
		Diff	-23	53	-5	-12	28	-54	13
		Rank	6	**1**	4	5	2	7	3
	PeakDist	Wins	51	13	35	10	27	33	0
		Losses	0	37	11	42	19	9	51
		Diff	51	-24	24	-32	8	24	-51
		Rank	**1**	5	2	6	4	2	7

Table B.4: Wins and losses for best algorithm configurations selected based on $offline_{Error}$ for various modalities (shift value of 0.05 for all cases)

Peaks	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
10	Error	Wins	35	20	29	8	0	5	17
		Losses	0	12	5	21	34	27	15
		Diff	35	8	24	-13	-34	-22	2
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	22	35	25	10	22	1	7
		Losses	12	1	10	25	12	34	28
		Diff	10	34	15	-15	10	-33	-21
		Rank	3	**1**	2	5	3	7	6
	PeakDist	Wins	34	14	23	2	24	17	3
		Losses	1	19	9	33	7	16	32
		Diff	33	-5	14	-31	17	1	-29
		Rank	**1**	5	3	7	2	4	6
30	Error	Wins	33	22	28	10	0	7	16
		Losses	0	11	5	19	35	27	19
		Diff	33	11	23	-9	-35	-20	-3
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	17	35	30	9	23	0	10
		Losses	17	1	6	27	13	36	24
		Diff	0	34	24	-18	10	-36	-14
		Rank	4	**1**	2	6	3	7	5
	PeakDist	Wins	33	13	21	3	30	16	2
		Losses	0	22	13	31	3	16	33
		Diff	33	-9	8	-28	27	0	-31
		Rank	**1**	5	3	6	2	4	7
50	Error	Wins	34	22	28	10	0	7	15
		Losses	0	11	5	19	35	27	19
		Diff	34	11	23	-9	-35	-20	-4
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	22	34	20	7	17	0	17
		Losses	12	0	13	27	16	36	13
		Diff	10	34	7	-20	1	-36	4
		Rank	2	**1**	3	6	5	7	4
	PeakDist	Wins	32	12	20	2	25	20	2
		Losses	1	24	11	32	3	10	32
		Diff	31	-12	9	-30	22	10	-30
		Rank	**1**	5	4	6	2	3	6
70	Error	Wins	33	19	25	8	0	7	13
		Losses	0	10	5	10	35	27	18
		Diff	33	9	20	-2	-35	-20	-5
		Rank	**1**	3	2	4	7	6	5
	PeakCover	Wins	21	35	25	7	17	0	16
		Losses	14	0	8	29	16	36	18
		Diff	7	35	17	-22	1	-36	-2
		Rank	3	**1**	2	6	4	7	5
	PeakDist	Wins	33	13	21	2	24	18	4
		Losses	1	23	9	33	5	12	32
		Diff	32	-10	12	-31	19	6	-28
		Rank	**1**	5	3	7	2	4	6
90	Error	Wins	32	25	27	10	0	7	14
		Losses	1	9	7	17	35	27	19
		Diff	31	16	20	-7	-35	-20	-5
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	20	36	21	8	18	0	19
		Losses	15	0	12	28	17	36	14
		Diff	5	36	9	-20	1	-36	5
		Rank	3	**1**	2	6	5	7	3
	PeakDist	Wins	32	14	22	1	21	22	3
		Losses	0	21	11	33	11	8	31
		Diff	32	-7	11	-32	10	14	-28
		Rank	**1**	5	3	7	4	2	6

Table B.5: Wins and losses for best algorithm configurations selected based on $offline_{PeakCover}$ for various modalities (shift value of 0.05 for all cases)

Peaks	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
10	Error	Wins	32	27	26	3	0	5	18
		Losses	2	5	6	24	31	25	18
		Diff	30	22	20	-21	-31	-20	0
		Rank	**1**	2	3	6	7	5	4
	PeakCover	Wins	24	28	36	6	12	0	18
		Losses	10	7	0	30	24	36	17
		Diff	14	21	36	-24	-12	-36	1
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	26	8	12	10	30	27	0
		Losses	7	23	19	24	2	2	36
		Diff	19	-15	-7	-14	28	25	-36
		Rank	3	6	4	5	**1**	2	7
30	Error	Wins	36	20	30	11	5	0	20
		Losses	0	14	6	24	29	35	14
		Diff	36	6	24	-13	-24	-35	6
		Rank	**1**	3	2	5	6	7	3
	PeakCover	Wins	23	29	36	6	12	0	20
		Losses	13	7	0	30	24	36	16
		Diff	10	22	36	-24	-12	-36	4
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	25	6	12	18	33	28	0
		Losses	9	30	24	18	1	4	36
		Diff	16	-24	-12	0	32	24	-36
		Rank	3	6	5	4	**1**	2	7
50	Error	Wins	36	18	30	9	7	0	22
		Losses	0	16	6	25	28	35	12
		Diff	36	2	24	-16	-21	-35	10
		Rank	**1**	4	2	5	6	7	3
	PeakCover	Wins	22	29	36	6	12	0	20
		Losses	13	6	0	30	24	36	16
		Diff	9	23	36	-24	-12	-36	4
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	24	6	12	16	30	32	0
		Losses	10	30	22	18	4	0	36
		Diff	14	-24	-10	-2	26	32	-36
		Rank	3	6	5	4	2	**1**	7
70	Error	Wins	35	18	30	9	7	0	22
		Losses	0	16	5	25	28	35	12
		Diff	35	2	25	-16	-21	-35	10
		Rank	**1**	4	2	5	6	7	3
	PeakCover	Wins	22	29	36	6	12	0	19
		Losses	12	6	0	30	24	36	16
		Diff	10	23	36	-24	-12	-36	3
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	24	5	12	17	32	30	0
		Losses	10	30	23	18	2	2	35
		Diff	14	-25	-11	-1	30	28	-35
		Rank	3	6	5	4	**1**	2	7
90	Error	Wins	35	21	29	8	7	0	20
		Losses	1	12	7	25	27	35	13
		Diff	34	9	22	-17	-20	-35	7
		Rank	**1**	3	2	5	6	7	4
	PeakCover	Wins	22	31	35	6	12	0	19
		Losses	13	5	1	30	24	36	16
		Diff	9	26	34	-24	-12	-36	3
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	23	11	12	15	29	32	0
		Losses	13	25	23	20	4	1	36
		Diff	10	-14	-11	-5	25	31	-36
		Rank	3	6	5	4	2	**1**	7

Table B.6: Wins and losses for best algorithm configurations selected based on $offline_{PeakDist}$ for various modalities (shift value of 0.05 for all cases)

Peaks	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
10	Error	Wins	34	24	25	1	0	9	18
		Losses	0	6	5	28	30	24	18
		Diff	34	18	20	-27	-30	-15	0
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	8	36	21	10	23	0	15
		Losses	25	0	9	21	8	36	14
		Diff	-17	36	12	-11	15	-36	1
		Rank	6	**1**	3	5	2	7	4
	PeakDist	Wins	34	11	25	6	21	18	0
		Losses	2	23	8	29	9	9	35
		Diff	32	-12	17	-23	12	9	-35
		Rank	**1**	5	2	6	3	4	7
30	Error	Wins	32	26	28	3	1	7	18
		Losses	2	8	5	28	30	25	17
		Diff	30	18	23	-25	-29	-18	1
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	8	36	15	12	22	2	23
		Losses	28	0	17	21	11	34	7
		Diff	-20	36	-2	-9	11	-32	16
		Rank	6	**1**	4	5	3	7	2
	PeakDist	Wins	33	11	23	7	21	20	0
		Losses	2	25	8	29	8	7	36
		Diff	31	-14	15	-22	13	13	-36
		Rank	**1**	5	2	6	3	3	7
50	Error	Wins	33	26	27	3	2	5	18
		Losses	2	8	6	26	31	25	16
		Diff	31	18	21	-23	-29	-20	2
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	7	36	14	14	25	0	27
		Losses	29	0	22	21	9	35	7
		Diff	-22	36	-8	-7	16	-35	20
		Rank	6	**1**	5	4	3	7	2
	PeakDist	Wins	33	10	24	7	20	22	0
		Losses	1	25	8	28	10	8	36
		Diff	32	-15	16	-21	10	14	-36
		Rank	**1**	5	2	6	4	3	7
70	Error	Wins	33	26	30	6	2	5	18
		Losses	2	10	6	25	33	27	17
		Diff	31	16	24	-19	-31	-22	1
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	7	36	14	15	24	0	28
		Losses	29	0	22	20	11	35	7
		Diff	-22	36	-8	-5	13	-35	21
		Rank	6	**1**	5	4	3	7	2
	PeakDist	Wins	34	10	25	7	19	24	0
		Losses	1	25	8	28	15	6	36
		Diff	33	-15	17	-21	4	18	-36
		Rank	**1**	5	3	6	4	2	7
90	Error	Wins	33	26	28	3	1	4	19
		Losses	3	9	6	25	29	26	16
		Diff	30	17	22	-22	-28	-22	3
		Rank	**1**	3	2	5	7	5	4
	PeakCover	Wins	7	33	14	14	26	0	30
		Losses	28	3	22	20	10	35	6
		Diff	-21	30	-8	-6	16	-35	24
		Rank	6	**1**	5	4	3	7	2
	PeakDist	Wins	33	8	23	7	18	23	0
		Losses	1	22	8	25	14	6	36
		Diff	32	-14	15	-18	4	17	-36
		Rank	**1**	5	3	6	4	2	7

Table B.7: Wins and losses for best algorithm configurations selected based on $offline_{Error}$ for various shift severities (Number of peaks set to 10 for all cases)

Shift	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
0.01	Error	Wins	35	23	30	8	0	6	17
		Losses	0	12	5	23	34	27	18
		Diff	35	11	25	-15	-34	-21	-1
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	21	35	25	10	23	0	8
		Losses	13	0	10	25	11	35	28
		Diff	8	35	15	-15	12	-35	-20
		Rank	4	**1**	2	5	3	7	6
	PeakDist	Wins	34	14	21	4	26	18	2
		Losses	1	19	11	32	7	15	34
		Diff	33	-5	10	-28	19	3	-32
		Rank	**1**	5	3	6	2	4	7
0.05	Error	Wins	35	23	26	11	0	8	16
		Losses	0	11	9	21	35	27	16
		Diff	35	12	17	-10	-35	-19	0
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	25	33	30	6	16	1	12
		Losses	10	1	5	29	19	35	24
		Diff	15	32	25	-23	-3	-34	-12
		Rank	3	**1**	2	6	4	7	5
	PeakDist	Wins	36	14	21	4	25	18	2
		Losses	0	20	13	31	8	16	32
		Diff	36	-6	8	-27	17	2	-30
		Rank	**1**	5	3	6	2	4	7
0.1	Error	Wins	35	19	25	12	0	7	17
		Losses	1	13	10	15	35	27	14
		Diff	34	6	15	-3	-35	-20	3
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	25	28	25	6	21	1	11
		Losses	7	4	8	30	10	35	23
		Diff	18	24	17	-24	11	-34	-12
		Rank	2	**1**	3	6	4	7	5
	PeakDist	Wins	35	13	22	1	26	19	4
		Losses	0	22	12	34	7	15	30
		Diff	35	-9	10	-33	19	4	-26
		Rank	**1**	5	3	7	2	4	6
0.3	Error	Wins	35	20	25	12	0	7	17
		Losses	1	12	9	17	35	27	15
		Diff	34	8	16	-5	-35	-20	2
		Rank	**1**	3	2	5	7	6	4
	PeakCover	Wins	31	26	27	5	16	1	13
		Losses	3	5	5	31	18	35	22
		Diff	28	21	22	-26	-2	-34	-9
		Rank	**1**	3	2	6	4	7	5
	PeakDist	Wins	36	14	22	1	26	19	6
		Losses	0	21	14	34	10	16	29
		Diff	36	-7	8	-33	16	3	-23
		Rank	**1**	5	3	7	2	4	6

Table B.8: Wins and losses for best algorithm configurations selected based on $offline_{PeakCover}$ for various shift severities (Number of peaks set to 10 for all cases)

Shift	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
0.01	Error	Wins	33	26	25	7	2	5	18
		Losses	2	4	6	25	32	29	18
		Diff	31	22	19	-18	-30	-24	0
		Rank	**1**	2	3	5	7	6	4
	PeakCover	Wins	24	29	35	6	12	0	18
		Losses	10	5	1	30	24	36	18
		Diff	14	24	34	-24	-12	-36	0
		Rank	3	2	**1**	6	5	7	4
	PeakDist	Wins	26	12	12	9	30	28	0
		Losses	8	21	23	25	2	2	36
		Diff	18	-9	-11	-16	28	26	-36
		Rank	3	4	5	6	**1**	2	7
0.05	Error	Wins	35	27	24	3	0	7	20
		Losses	0	7	11	27	30	25	16
		Diff	35	20	13	-24	-30	-18	4
		Rank	**1**	2	3	6	7	5	4
	PeakCover	Wins	30	26	28	7	15	0	11
		Losses	3	6	3	29	18	36	22
		Diff	27	20	25	-22	-3	-36	-11
		Rank	**1**	3	2	6	4	7	5
	PeakDist	Wins	32	8	15	11	25	28	2
		Losses	2	26	21	24	10	4	34
		Diff	30	-18	-6	-13	15	24	-32
		Rank	**1**	6	4	5	3	2	7
0.1	Error	Wins	34	25	28	9	3	0	18
		Losses	0	10	6	24	28	32	17
		Diff	34	15	22	-15	-25	-32	1
		Rank	**1**	3	2	5	6	7	4
	PeakCover	Wins	34	26	25	7	20	0	10
		Losses	1	10	9	28	14	36	24
		Diff	33	16	16	-21	6	-36	-14
		Rank	**1**	2	2	6	4	7	5
	PeakDist	Wins	33	6	20	11	20	24	1
		Losses	0	27	13	25	12	5	33
		Diff	33	-21	7	-14	8	19	-32
		Rank	**1**	6	4	5	3	2	7
0.3	Error	Wins	36	26	28	9	5	0	17
		Losses	0	10	8	24	28	33	18
		Diff	36	16	20	-15	-23	-33	-1
		Rank	**1**	3	2	5	6	7	4
	PeakCover	Wins	35	22	25	7	21	0	11
		Losses	1	11	10	28	12	36	23
		Diff	34	11	15	-21	9	-36	-12
		Rank	**1**	3	2	6	4	7	5
	PeakDist	Wins	32	8	18	11	22	28	1
		Losses	1	27	16	25	13	3	35
		Diff	31	-19	2	-14	9	25	-34
		Rank	**1**	6	4	5	3	2	7

Table B.9: Wins and losses for best algorithm configurations selected based on $offline_{PeakDist}$ for various shift severities (Number of peaks set to 10 for all cases)

Shift	Metric	Data	AMSO	ANPSO	CPSOR	MCPSO	MQSO	SAMO	SPSO
0.01	Error	Wins	31	24	26	1	3	7	18
		Losses	0	6	3	29	29	25	18
		Diff	31	18	23	-28	-26	-18	0
		Rank	**1**	3	2	7	6	5	4
	PeakCover	Wins	8	33	26	10	23	0	18
		Losses	27	2	9	23	10	35	12
		Diff	-19	31	17	-13	13	-35	6
		Rank	6	**1**	2	5	3	7	4
	PeakDist	Wins	34	13	25	7	20	17	0
		Losses	2	20	8	29	10	11	36
		Diff	32	-7	17	-22	10	6	-36
		Rank	**1**	5	2	6	3	4	7
0.05	Error	Wins	32	22	29	1	3	7	19
		Losses	1	9	5	30	29	24	15
		Diff	31	13	24	-29	-26	-17	4
		Rank	**1**	3	2	7	6	5	4
	PeakCover	Wins	15	35	24	10	26	0	8
		Losses	16	0	11	23	8	36	24
		Diff	-1	35	13	-13	18	-36	-16
		Rank	4	**1**	3	5	2	7	6
	PeakDist	Wins	32	12	19	7	20	23	0
		Losses	0	22	11	28	12	4	36
		Diff	32	-10	8	-21	8	19	-36
		Rank	**1**	5	3	6	3	2	7
0.1	Error	Wins	34	23	29	2	1	10	20
		Losses	1	11	7	29	31	25	15
		Diff	33	12	22	-27	-30	-15	5
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	14	35	20	8	28	0	10
		Losses	15	1	11	25	5	36	22
		Diff	-1	34	9	-17	23	-36	-12
		Rank	4	**1**	3	6	2	7	5
	PeakDist	Wins	31	9	17	10	22	25	0
		Losses	0	26	14	26	9	4	35
		Diff	31	-17	3	-16	13	21	-35
		Rank	**1**	6	4	5	3	2	7
0.3	Error	Wins	35	22	27	1	1	11	20
		Losses	0	12	6	30	31	24	14
		Diff	35	10	21	-29	-30	-13	6
		Rank	**1**	3	2	6	7	5	4
	PeakCover	Wins	21	32	17	6	28	0	12
		Losses	12	1	15	27	4	35	22
		Diff	9	31	2	-21	24	-35	-10
		Rank	3	**1**	4	6	2	7	5
	PeakDist	Wins	32	7	17	8	19	26	0
		Losses	1	24	13	25	10	2	34
		Diff	31	-17	4	-17	9	24	-34
		Rank	**1**	5	4	5	3	2	7